Joakim Edvardsen

Assignment 5 - Docker & Kubernetes

Docker concepts

Images

An images is a file that or template with instruction for creating a docker container.

Containers

A container is a runnable instance of an image. The containers can be created, started, stopped, moved and deleted. The idea is that you define a container via an image and then you can easely spin up containers to run your application.

Task 2 - Create a container

Created a basic spring-boot web application with a "Hello world" endpoint at `/hello`

Created a Dockerfile that uses `amazoncorretto`, takes the .jar file created whe building the spring-boot and copies it into the container before running `java -jar app.jar` to run it.

Heres the Dockerfile:

```
FROM amazoncorretto:19.0.1-alpine

WORKDIR /app

COPY target/*.jar app.jar

ENTRYPOINT ["java","-jar","app.jar"]
```

To run it, first create the image: (creates image base on current directory)

```
docker build -t jkm00/spring-boot .
```

Second, create a container based on the image:

```
docker run -p 8080:8080 jkm00/spring.boot
```

jkm00 is my username on docker hub, replace this with your username (or simply remove it)

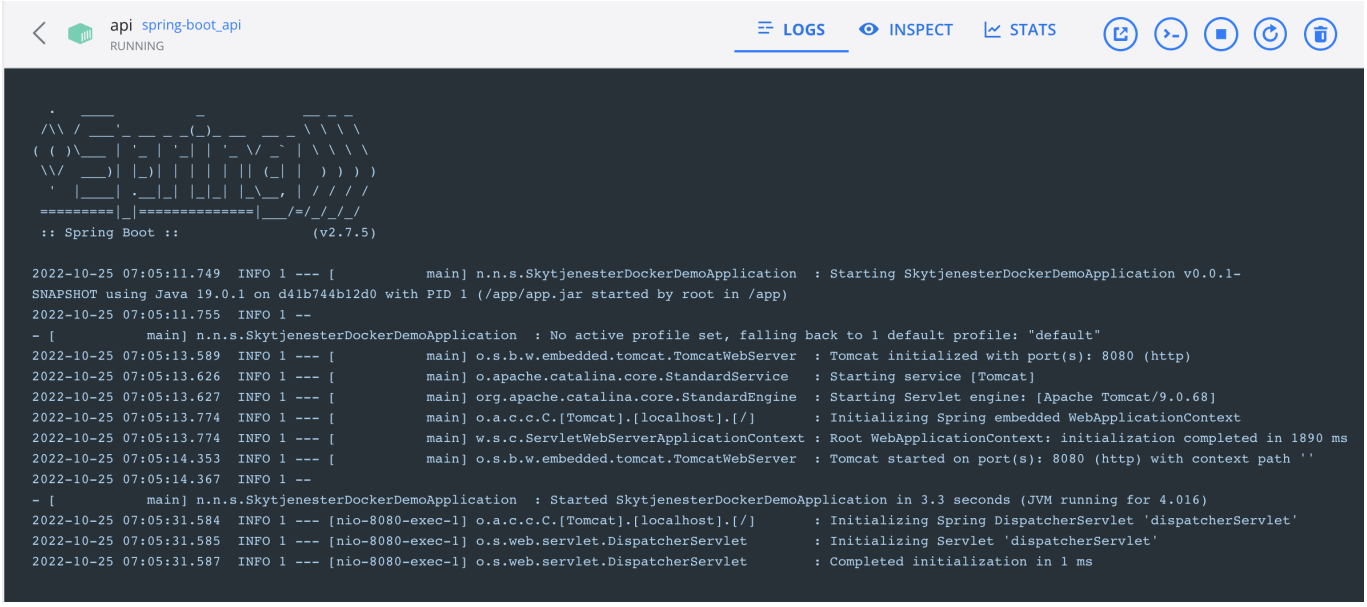
Also created a simple dokcer-compose that uses the Dockerfile above when spinning ut a container. Heres the file:

```
version: "3"
services:
  api:
    container_name: api
    build:
      context: ./
      dockerfile: Dockerfile
    ports:
      - 80:8080
```

To run this, use:

```
docker-compose up -d
```

Image of running container (with docker-compose):



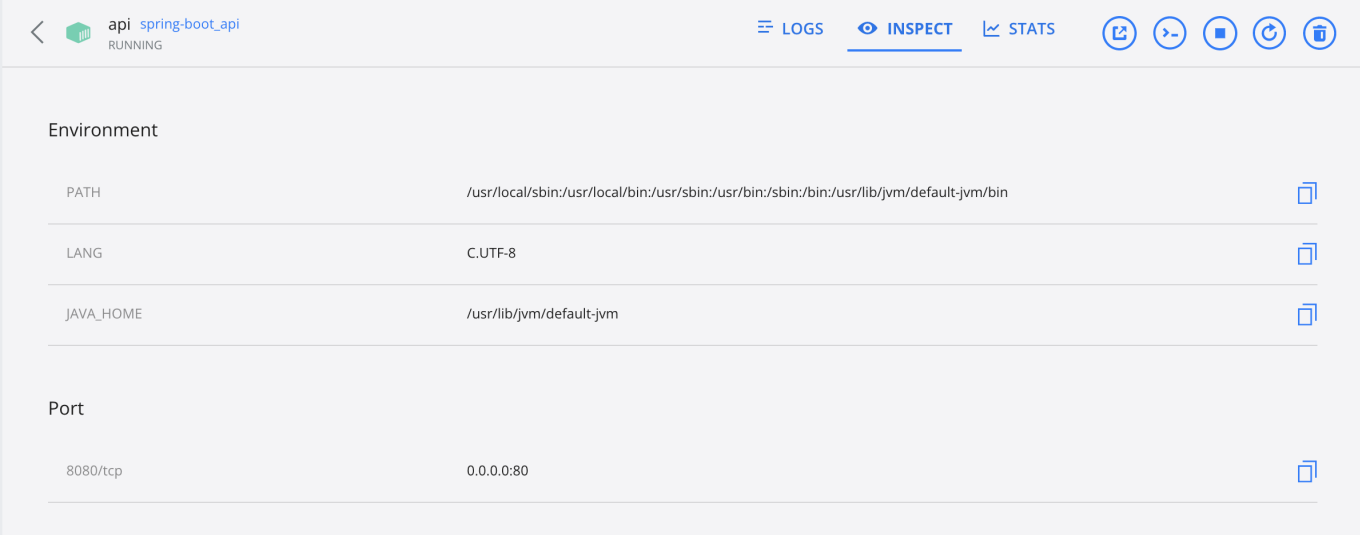
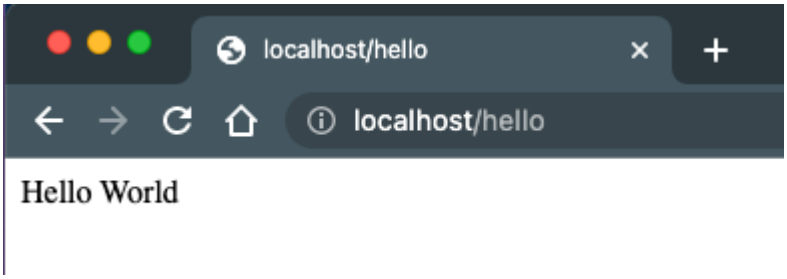


Image of accessing `/hello` endpoint:



Task 3 - Create a Docker Compose with a minimume of two services

For this I used the same basic spring-boot application from above, but this time I connected it to a PostgreSQL database.

For this I had to create a docker-compose file that firstly boots ut the PostgreSQL container, and when this container is up and running only then will the container with the spring-boot application spin up.

Heres the docker-compose file:

```
version: "3.5"
services:
  api:
    container_name: api
    image: jkm00/spring-boot
    ports:
      - 80:8080
    environment:
      - "SPRING_PROFILE_ACTIVE=prod"
      - POSTGRES_PORT=${POSTGRES_PORT}
      - POSTGRES_DB=${POSTGRES_DB}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    # Make sure database is up and running before starting the api
    depends_on:
      db:
        condition: service_healthy
```

```
  db:
    container_name: database
    image: postgres:12.2
    restart: always
    environment:
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
    ports:
      - ${POSTGRES_PORT}:5432
    healthcheck:
      test: ["CMD-SHELL", "pg_isready"]
      interval: 10s
      timeout: 5s
      retries: 5
```

Note the `depends_on` attribute in the `api` definition

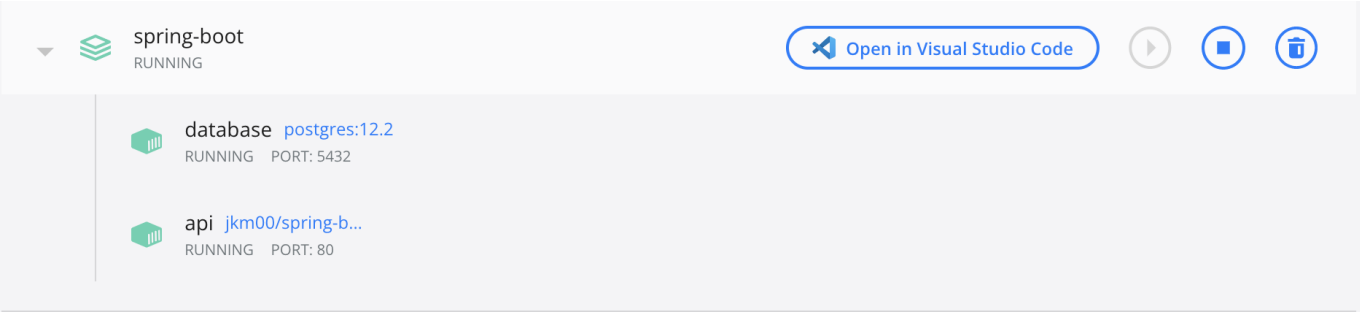
I used the same docker images for the spring-boot as I used in the task 2.

Because this file is usually a file stored in a git repository, you dont want to expose all the database credentials. Thats why I've used environment variables. These variables are stored in a local `.env` file that is not pushed to the repository. Docker will automatically read any file with the `.env` extension and populate the variables in the docker-compose file.

Heres an example of the `.env` file:

```
POSTGRES_NAME=somename
POSTGRES_USER=someusername
POSTGRES_PASSWORD=asecretpassword
POSTGRES_PORT=5432
POSTGRES_DB=somedbname
```

Images of running containers:



database postgres:12.2
RUNNING

LOGS INSPECT STATS

Environment

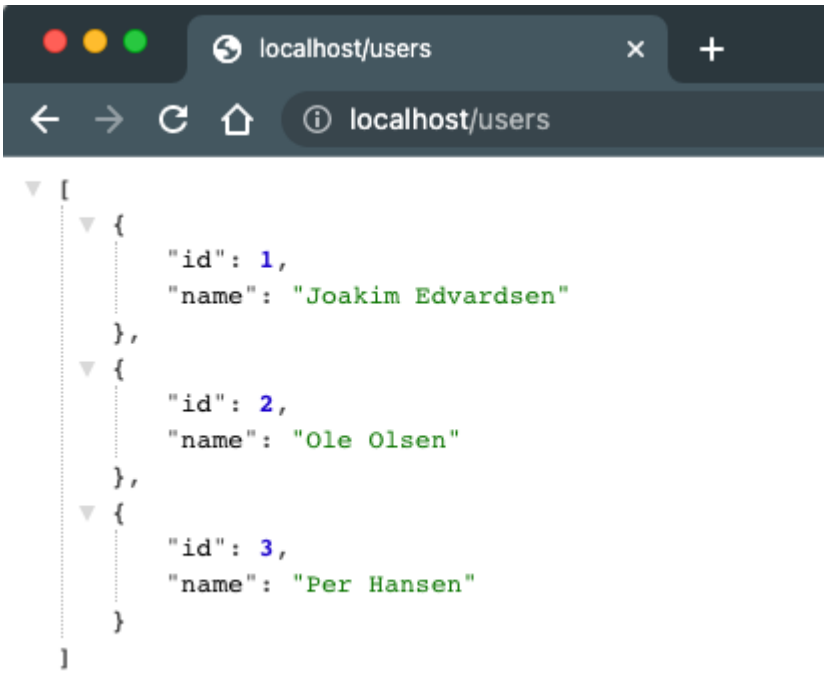
POSTGRES_USER	jkm
POSTGRES_PASSWORD	mysecretpassword
POSTGRES_DB	skytjenester
PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/postgresql/12/bin
GOSU_VERSION	1.12
LANG	en_US.utf8
PG_MAJOR	12
PG_VERSION	12.2-2.pgdg100+1
PGDATA	/var/lib/postgresql/data

api jkm00/spring-boot
RUNNING

LOGS INSPECT STATS

```
- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.6.12.Final
2022-10-25 11:45:33.761  INFO 1 --
- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-10-25 11:45:34.012  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2022-10-25 11:45:34.261  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2022-10-25 11:45:34.487  INFO 1 --
- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQL10Dialect
2022-10-25 11:45:35.845  INFO 1 --
- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.t
ransaction.jta.platform.internal.NoJtaPlatform]
2022-10-25 11:45:35.863  INFO 1 --
- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-10-25 11:45:36.509  WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-
view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.ope
n-in-view to disable this warning
2022-10-25 11:45:37.123  INFO 1 --
- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-25 11:45:37.138  INFO 1 --
- [           main] n.n.s.SkytjenesterDockerDemoApplication  : Started SkytjenesterDockerDemoApplication in 8.667 seconds (JVM runn
ing for 9.786)
2022-10-25 11:45:37.226  INFO 1 --- [           main] n.n.s.DummyDataInitializer               : Importing dummy data...
2022-10-25 11:45:37.317  INFO 1 --- [           main] n.n.s.DummyDataInitializer               : Finished initializing dummy data!
2022-10-25 11:52:06.302  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].
[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-10-25 11:52:06.304  INFO 1 --- [nio-8080-exec-
1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-10-25 11:52:06.326  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 21 ms
```

Result of accessing `/users` endpoint:



Docker vs Kubernetes

- Docker is a way of containerizing your application, meaning creating an isolated environment for the application.
- Kubernetes is a way of structuring the containers. Grouping the containers that make up an application in a cluster.
- Docker is great for automatic building and deployment (Before for and during deployment)
- Kubernetes is great for scheduling and managing the containers after deployment.
- Kubernetes has support for automatic scaling and monitoring

Sources

- Docker vs Kubernetes
- Docker documentation