DevOps Rapport
Celine Pöhl

TP 01 – 23/10/23

Database

```
(base) [Celine♥♥♥♥♥Database]$docker build -t celine99/tddatabase .
[+] Building 11.1s (6/6) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.0s
 => => transferring dockerfile: 133B                                              0.0s
 => [internal] load .dockerignore                                                 0.0s
 => => transferring context: 2B                                                   0.0s
 => [internal] load metadata for docker.io/library/postgres:14.1-alpine           1.8s
 => [auth] library/postgres:pull token for registry-1.docker.io                   0.0s
 => [1/1] FROM docker.io/library/postgres:14.1-alpine@sha256:578ca5c8452c08a4e0f5  8.9s
 => => resolve docker.io/library/postgres:14.1-alpine@sha256:578ca5c8452c08a4e0f5  0.0s
 => => sha256:578ca5c8452c08a4e0f5e65b55dce5e1812fe63c8fee40ea837641031598e51e 1.65kB / 1.65kB  0.0s
 => => sha256:884c142deb4a141f67489c807534ec6139f13b9a6432d2f87a4de283aaec0b5c 1.99kB / 1.99kB  0.0s
 => => sha256:1149d285a5f5c4340cefa2211869c3a6b1128ac78974545e0b4fe62d3d0e66a8 7.84kB / 7.84kB  0.0s
 => => sha256:59bf1c3509f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3 2.82MB / 2.82MB  0.3s
 => => sha256:c50e01d57241cf7ef93a91060f5eb0b895a4b443f20dc1ce5e77d441184a6dc2 1.28kB / 1.28kB  0.2s
 => => sha256:a0646b0f1eadaf0cd3fdb4c4490a69c4c7aed9b7ae10b24eb9005c59aa0b6e57 148B / 148B      0.2s
 => => sha256:7433e5151e0cee31a0d5b90433751e809eb3b860b250f73f0720f7d05788dc 78.90MB / 78.90MB  2.6s
 => => extracting sha256:59bf1c3509f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3       0.3s
 => => sha256:8de463f7fd190f136a6f5cf50d5d8add707fd78d079d6391d952a02dcd40e412 162B / 162B      0.5s
 => => sha256:8854018388d9035028f41a2c0494acf2868e219888840026df2227cc4728a8d 9.20kB / 9.20kB  0.5s
 => => sha256:b39ee18abab98838412adb823af56cad2e732c25ac47da3ed4af92e41dbd2a7f 194B / 194B      0.7s
 => => sha256:11d7473a0ff973b03640db89b7278d2694eb298a52a84ad71b4e14a1ce3de45f 4.72kB / 4.72kB  0.7s
 => => extracting sha256:c50e01d57241cf7ef93a91060f5eb0b895a4b443f20dc1ce5e77d441184a6dc2       0.0s
 => => extracting sha256:a0646b0f1eadaf0cd3fdb4c4490a69c4c7aed9b7ae10b24eb9005c59aa0b6e57       0.0s
 => => extracting sha256:7433e5151e0cee31a0d5b90433751e809eb3b860b250f73f0720f7d05788dc34       5.7s
 => => extracting sha256:8854018388d9035028f41a2c0494acf2868e219888840026df2227cc4728a8d       0.0s
 => => extracting sha256:8de463f7fd190f136a6f5cf50d5d8add707fd78d079d6391d952a02dcd40e412       0.0s
 => => extracting sha256:b39ee18abab98838412adb823af56cad2e732c25ac47da3ed4af92e41dbd2a7f       0.0s
 => => extracting sha256:11d7473a0ff973b03640db89b7278d2694eb298a52a84ad71b4e14a1ce3de45f       0.0s
 => exporting to image                                                            0.0s
 => => exporting layers                                                           0.0s
 => => writing image sha256:6685d0c4833878dbaf36a3d7be0ba0cd1b87bd37dd22ad8d74ef42391e614a48    0.0s
 => => naming to docker.io/celine99/tddatabase                                    0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```

```
(base) [Celine♥♥♥♥♥Database]$docker run -d -p 8888:5432 --name tddatabase celine99/tddatabase
753aac0aa3b03ea9c814bbea4163275fe2b3a260f90f5fe2bfcbe3e8041c7b3d
```

Rerun:

```
(base) [Celine♥♥♥♥♥Database]$docker run -d  --network app-network  --name tddatabase celine99/tddat
abase
```

Why should we run the container with a flag `-e` to give the environment variables?

Storing passwords in plain text is a security risk because if an attacker gains access to your Docker image, they can easily extract the sensitive information.

I changed the Dockerfile to:

```
FROM POSTGRES:14.1-ALPINE

ENV POSTGRES_USER=USR
ENV POSTGRES_DB=DB
```

and then:

```
(base) [Celine♥♥♥♥♥Database]$docker run -d  --network app-network -e POSTGRES_PASSWOrD=pwd --name t
ddatabase celine99/tddatabase
```

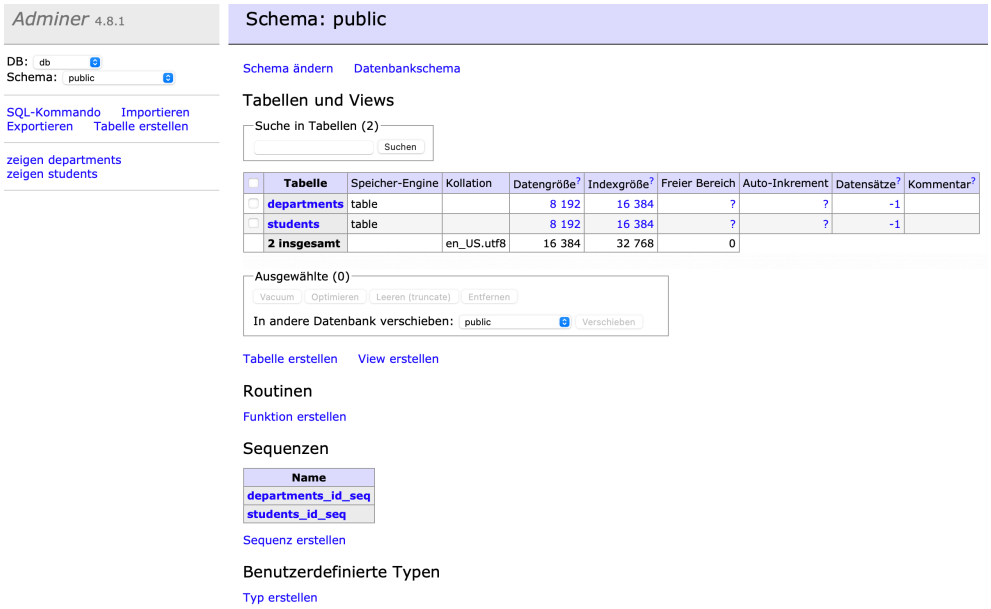
```
(base) [Celine♥♥♥♥♥Database]$docker network create app-network
e2734f052e22df2005dd0247a14915fb6445910c063a79ea306ac27f890b272c
```

```
(base) [Celine♥♥♥♥♥DevOps]$    docker run \
>     -p "8090:8080" \
>     --net=app-network \
>     --name=adminer \
>     -d \
>     adminer
Unable to find image 'adminer:latest' locally
latest: Pulling from library/adminer
69b3efbf67c2: Pull complete
3a0fdd089fb3: Pull complete
689786b0c396: Pull complete
6bcf42cdb6b5: Pull complete
d507b34ef5ac: Pull complete
04b8d4f964a8: Pull complete
7a75f8703cd7: Pull complete
Digest: sha256:f762276d79d2f18ae7fe28c79ee25a0a3b3dba9dc92ed695a3e88d613b3e6bde
Status: Downloaded newer image for adminer:latest
ca0a22e552e97e97aab87115ea440097dc67e3f8a55c015754bfb5bb5d96aafb
```

I created the folder "sqlfiles" with the two sql files. In my Dockerfile, I added the following:

```
1 FROM postgres:14.1-alpine
2
3 ENV      POSTGRES_DB=db \
4          POSTGRES_USER=usr
5
6 COPY sqlfiles/01-CreateScheme.sql /docker-entrypoint-initdb.d/
7 COPY sqlfiles/02-InsertData.sql /docker-entrypoint-initdb.d/
```

I rebuilt my image. Then I could see the data in my database.



Persist data

```
(base) [Celine♥♥♥♥♥Database]$docker volume create myDatadir
myDatadir
```

```
(base) [Celine♥♥♥♥♥Database]$docker run -d  --network app-network -e POSTGRES_PASSWOrD=pwd -v myDat
adir:/var/lib/postgresql/data --name tddatabase celine99/tddatabase
849b8a23aa8af375d626d2638348825ade60b6e1442a7a082f4633ba7fb60d4f
```

Backend API

I copied the Hello World Java file, compiled it and I have written the following Docker file:

```
1 FROM openjdk:11-jre-slim
2
3 WORKDIR /app
4
5 COPY Main.class /app
6
7 CMD ["java", "Main"]
```

```
(base) [Celine♥♥♥♥♥BackendAPI]$docker build -t celine99/tdbackend .
[+] Building 1.5s (9/9) FINISHED                                                docker:desktop-linux
 => [internal] load .dockerignore                                                              0.0s
 => => transferring context: 2B                                                                0.0s
 => [internal] load build definition from Dockerfile                                           0.0s
 => => transferring dockerfile: 120B                                                           0.0s
 => [internal] load metadata for docker.io/library/openjdk:11-jre-slim                         1.2s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                 0.0s
 => [1/3] FROM docker.io/library/openjdk:11-jre-slim@sha256:93af7df2308c5141a751c4830e6b6c5717  0.0s
 => [internal] load build context                                                              0.0s
 => => transferring context: 453B                                                              0.0s
 => CACHED [2/3] WORKDIR /app                                                                  0.0s
 => [3/3] COPY Main.class /app                                                                 0.0s
 => exporting to image                                                                         0.0s
 => => exporting layers                                                                        0.0s
 => => writing image sha256:c913275ac5be69ad9b0c89e2f4c70b5ce4094b3f5cc25f427a5d5b350a2fbdb1   0.0s
 => => naming to docker.io/celine99/tdbackend                                                  0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
(base) [Celine♥♥♥♥♥BackendAPI]$docker run --name tdbackend celine99/tdbackend
Hello World!
```

Multistage build

```
(base) [Celine♥♥♥♥♥BackendAPI]$docker build -t celine99/tdbackend .
[+] Building 62.2s (14/14) FINISHED                                             docker:desktop-linux
```

```
(base) [Celine♥♥♥♥♥BackendAPI]$docker run --name tdbackend celine99/tdbackend

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.17)
```

1-2 Why do we need a multistage build? And explain each step of this dockerfile.

In the **#build** section, we are setting up a build environment with Maven and Amazon Corretto

FROM maven:3.8.6-amazoncorretto-17 AS myapp-build:
specifies the base image for the build stage. We are using an image that includes Maven and Amazon Corretto. The **AS myapp-build** labels this stage with the name "myapp-build."

ENV MYAPP_HOME /opt/myapp: This line sets the environment variable **MYAPP_HOME** to **/opt/myapp**. It's used as the base directory for our application.

WORKDIR $MYAPP_HOME: This sets the working directory inside the container to the value of the **MYAPP_HOME** environment variable.

COPY pom.xml .: This copies the project's **pom.xml** file into the container.

COPY src ./src: This copies the source code from the local directory into the container's working directory.

RUN mvn package -DskipTests: This command runs Maven to build our Java application. The **-DskipTests** option skips running tests during the build process.

In the **#run** section, we are creating the runtime environment for our application and copying the built JAR file from the previous build stage.

FROM amazoncorretto:17: This line specifies the base image for the runtime stage.

ENV MYAPP_HOME /opt/myapp: This line sets the environment variable **MYAPP_HOME** to the same path as in the build stage.

WORKDIR $MYAPP_HOME: This sets the working directory inside the container to the value of the **MYAPP_HOME** environment variable.

COPY --from=myapp-build $MYAPP_HOME/target/*.jar $MYAPP_HOME/myapp.jar: This copies the JAR file built in the previous build stage from the build stage to the runtime stage.

ENTRYPOINT java -jar myapp.jar: This sets the command that is run when the container starts. It runs your Java application using the **java -jar myapp.jar** command.

The multi-stage build allows you to separate the heavier build environment (Maven) from the smaller runtime environment (Java). This results in a more efficient and smaller final Docker image, containing only what is necessary to run our application.

I changed the application.yml file to:

```yaml
spring:
  jpa:
    properties:
      hibernate:
        jdbc:
          lob:
            non_contextual_creation: true
    generate-ddl: false
    open-in-view: true
  datasource:
    url: jdbc:postgresql://tddatabase:5432/db
    username: usr
    password: pwd
    driver-class-name: org.postgresql.Driver
management:
 server:
   add-application-context-header: false
 endpoints:
   web:
     exposure:
       include: health,info,env,metrics,beans,configprops
```

```
(base) [Celine♥♥♥♥♥BackendAPI]$docker build -t celine99/tdbackend .
[+] Building 12.5s (9/9) FINISHED                                  docker:desktop-linux
```

```
(base) [Celine♥♥♥♥♥BackendAPI]$docker run -d --network app-network -p 8080:8080 --name tdbackend ce
line99/tdbackend
```

And then, the backend works:

```
[{"id":1,"firstname":"Eli","lastname":"Copter","department":{"id":1,"name":"IRC"}}]
```

Http Server

```
(base) [Celine♥♥♥♥♥HTTPserver]$docker run -d --network app-network -p 8000:80 --name tdhttp celine9
9/tdhttp
dd938a786041aa71912cce1424f97eb3a53d678c7fef20064b779f68c2e80afe
```

```
FROM httpd:2.4

COPY ./index.html /usr/local/apache2/htdocs/

EXPOSE 80
```

Hallo Welt!

to retrieve this default configuration from your running container:

```
(base) [Celine♥♥♥♥♥BackendAPI]$docker cp tdhttp:/usr/local/apache2/conf/httpd.conf .
                                                Successfully copied 22.5kB to /Users/celine/Documents/
DevOps/BackendAPI/.
```

Reverse proxy
I added the part of the tutorial to the httpd.conf file and then changed the Dockerfile to always replace the httpd.conf file with my local version:

```
1 FROM httpd:2.4
2
3 COPY ./index.html /usr/local/apache2/htdocs/
4
5 COPY ./httpd.conf /usr/local/apache2/conf/httpd.conf
6
7 EXPOSE 80
```

```
(base) [Celine♥♥♥♥♥HTTPserver]$docker run -d --network app-network -p 8000:80 --name tdhttp celine9
9/tdhttp
05a80cc7622fc787d5e460b0a11853f55e5e3b416bb982d621d7e3243f7500b0
```

```
(base) [Celine♥♥♥♥♥HTTPserver]$docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED             STATUS
 PORTS                    NAMES
05a80cc7622f   celine99/tdhttp       "httpd-foreground"       2 minutes ago       Up About a minute
 0.0.0.0:8000->80/tcp     tdhttp
90c8b5bc0288   celine99/tdbackend    "/bin/sh -c 'java -j…"   About an hour ago   Up About an hour
 0.0.0.0:8080->8080/tcp   tdbackend
75f05031229f   celine99/tddatabase   "docker-entrypoint.s…"   About an hour ago   Up About an hour
 0.0.0.0:5432->5432/tcp   tddatabase
13d9c8de2b9e   adminer               "entrypoint.sh php -…"   20 hours ago        Up 20 hours
 0.0.0.0:8090->8080/tcp   adminer
```

Why do we need a reverse proxy?

A reverse proxy is needed to improve security and performance by serving as an intermediary between clients and backend servers. It helps hide server details.

Link application
docker-compose.yml file:

```yaml
version: '3.7'

services:
    backend:
        build: ./BackendAPI/
        container_name: tdbackend
        networks:
            - my-network
        depends_on:
            - database

    database:
        build: ./Database/
        container_name: tddatabase
        networks:
            - my-network

    httpd:
        build: ./HTTPserver/
        ports:
            - "8080:80"
        networks:
            - my-network
        depends_on:
            - backend

networks:
    my-network:
```

Why is docker-compose so important?

Docker Compose is important because it simplifies the deployment and management of multi-container Docker applications. It allows you to define, configure, and run multiple containers as a single application, making it easier to develop, test, and deploy applications.

1-3 Document docker-compose most important commands.

- **docker-compose up**: Starts the services defined in the **docker-compose.yml** file.
- **docker-compose down**: Stops and removes the containers defined in the **docker-compose.yml** file.
- **docker-compose build**: Builds or rebuilds the container images based on the Dockerfiles specified in the **docker-compose.yml** file.

1-4 Document your docker-compose file.

```
VERSION: '3.7'

SERVICES:
    BACKEND:
        BUILD: ./BACKENDAPI/    # BUILD THE BACKENDAPI SERVICE FROM THE SPECIFIED DIRECTORY.
        CONTAINER_NAME: TDBACKEND  # SET THE CONTAINER NAME FOR THE BACKEND SERVICE.
        NETWORKS:
            - MY-NETWORK  # CONNECT THE SERVICE TO THE 'MY-NETWORK' DOCKER NETWORK.
        DEPENDS_ON:
            - DATABASE  # ENSURE THAT THE BACKEND SERVICE STARTS AFTER THE 'DATABASE' SERVICE.
```

```
    DATABASE:
        BUILD: ./DATABASE/  # BUILD THE DATABASE SERVICE FROM THE SPECIFIED DIRECTORY.
        CONTAINER_NAME: TDDATABASE  # SET THE CONTAINER NAME FOR THE DATABASE SERVICE.
        NETWORKS:
            - MY-NETWORK  # CONNECT THE SERVICE TO THE 'MY-NETWORK' DOCKER NETWORK.

    HTTPD:
        BUILD: ./HTTPSERVER/  # BUILD THE HTTPSERVER SERVICE FROM THE SPECIFIED DIRECTORY.
        PORTS:
            - "8080:80"  # MAP PORT 8080 ON THE HOST TO PORT 80 IN THE CONTAINER.
        NETWORKS:
            - MY-NETWORK  # CONNECT THE SERVICE TO THE 'MY-NETWORK' DOCKER NETWORK.
        DEPENDS_ON:
            - BACKEND  # ENSURE THAT THE HTTPD SERVICE STARTS AFTER THE 'BACKEND' SERVICE.

NETWORKS:
    MY-NETWORK:  # DEFINE A DOCKER NETWORK NAMED 'MY-NETWORK' FOR SERVICE COMMUNICATION.
```

Publish

1-5 Document your publication commands and published images in dockerhub.

```
(base) [Celine❤️❤️❤️❤️❤️DevOps]$docker tag celine99/tdhttp celine99/tdhttp:1.0
(base) [Celine❤️❤️❤️❤️❤️DevOps]$docker push celine99/tdhttp:1.0
The push refers to repository [docker.io/celine99/tdhttp]
94e8f48c7b10: Pushed
2e11e1a8b136: Pushed
b7094d4685d5: Mounted from library/httpd
87ca57c6f4e9: Mounted from library/httpd
1343ea427053: Mounted from library/httpd
8db3e477577e: Mounted from library/httpd
cb4596cc1454: Mounted from library/httpd
1.0: digest: sha256:b544438f88173f420400d378e47b7f67f9ae0f7a63fd7d0ef6e297afcd2e9ea2 size: 1781
(base) [Celine❤️❤️❤️❤️❤️DevOps]$docker tag celine99/tdbackend celine99/tdbackend:1.0
(base) [Celine❤️❤️❤️❤️❤️DevOps]$docker push celine99/tdbackend:1.0
The push refers to repository [docker.io/celine99/tdbackend]
9c38e417bb6b: Pushed
d84f0131e215: Pushed
491e847f9b68: Mounted from library/amazoncorretto
ab18cb8eb197: Mounted from library/amazoncorretto
1.0: digest: sha256:4617c9d89f2ef83eb077bb3d169b3f513d85784bd3edfecd2222c81aec22801d size: 1161
(base) [Celine❤️❤️❤️❤️❤️DevOps]$docker tag celine99/tddatabase celine99/tddatabase:1.0
(base) [Celine❤️❤️❤️❤️❤️DevOps]$docker push celine99/tddatabase:1.0
The push refers to repository [docker.io/celine99/tddatabase]
693b2eb48bec: Pushed
a815ccfe95e3: Pushed
5b87e9731513: Mounted from library/postgres
176b9203da6e: Mounted from library/postgres
efb18f6577c9: Mounted from library/postgres
6c651825e7c4: Mounted from library/postgres
be6c168b4af5: Mounted from library/postgres
b737c2580132: Mounted from library/postgres
6cab14f8a434: Mounted from library/postgres
8d3ac3489996: Mounted from library/postgres
1.0: digest: sha256:04143b9d3c260a2d5f8b096926d2ae179ae9933605b4b47d23e73a7a2bab908a size: 2399
```