Dystrybucja oraz wersja systemu Linux wykorzystana do wdrożenia:

- Proxy: Debian GNU/Linux 12
- App: Red Hat Enterprise Linux 8.8 (Ootpa)
- DB: Ubuntu 22.04.3 LTS

Lista pakietów, które muszą być zainstalowane w systemie aby dokonać wdrożenia (lista nie musi uwzględniać pakietów-zależności, które zostaną automatycznie zainstalowane)

- curl
- ca-certificates
- java-17-openjdk-headless

Jeżeli w systemie użyto także oprogramowania spoza repozytoriów danej dystrybucji - opis czynności prowadzących do jego poprawnego zainstalowania

- Brak

Lokalizacje, w których powinny być umieszczone pliki dostarczone w celu wdrożenia

- Proxy:
  - Pliki konfiguracyjne nginx (nginx.conf) -> /etc/nginx/snippets/
  - Pliki związane z certyfikatem (self-signed.crt, self-signed.key, self-signed.conf, ssl-params.conf, dhparam.pem) -> /etc/ssl/certs/, /etc/ssl/private
- App:
  - Pliki związane z wdrożeniem (run-java.sh, ssbd06-runner.jar) -> /deployments
- DB:
  - Plik związany z inicjalizacją bazy danych (docker-init.sql) -> /docker-entrypoint-initdb.d/

Informację o zależnościach między kontenerami, które determinują kolejność ich uruchamiania

1. DB: musi zostać uruchomiony jako pierwszy ponieważ kontener z aplikacją musi mieć dostę do uruchomionej bazy danych.
2. App: uruchomiony jako drugi po uruchomienu bazy dancyh
3. Proxy: uruchamiany jako ostatni tak żeby aplikacja uruchamiana w 2 kroku była już dostępna

Zawartość plików opisujących konfiguracje kontenerów / zestawów kontenerów, o ile zostały wykorzystane

- Proxy:
  - Dockerfile

```
FROM nginx:latest

COPY nginx.conf /etc/nginx/

COPY self-signed.crt /etc/ssl/certs/self-signed.crt
COPY self-signed.key /etc/ssl/private/self-signed.key
COPY self-signed.conf /etc/nginx/snippets/self-signed.conf
COPY ssl-params.conf /etc/nginx/snippets/ssl-params.conf
COPY dhparam.pem /etc/ssl/certs/dhparam.pem

EXPOSE 80
```

- App:
  - Dockerfile

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.1

ARG JAVA_PACKAGE=java-17-openjdk-headless
ARG RUN_JAVA_VERSION=1.3.8

ENV LANG='en_US.UTF-8' LANGUAGE='en_US:en'

RUN microdnf install curl ca-certificates ${JAVA_PACKAGE} \
    && microdnf update \
    && microdnf clean all \
    && mkdir /deployments \
    && chown 1001 /deployments \
    && chmod "g+rwX" /deployments \
    && chown 1001:root /deployments \
    && curl https://repo1.maven.org/maven2/io/fabric8/run-java-
sh/${RUN_JAVA_VERSION}/run-java-sh-${RUN_JAVA_VERSION}-sh.sh -o
/deployments/run-java.sh \
    && chown 1001 /deployments/run-java.sh \
    && chmod 540 /deployments/run-java.sh \
    && echo "securerandom.source=file:/dev/urandom" >>
/etc/alternatives/jre/lib/security/java.security

ENV JAVA_OPTIONS="-Dquarkus.http.host=0.0.0.0 -
Djava.util.logging.manager=org.jboss.logmanager.LogManager"

COPY target/*-runner.jar /deployments/app.jar

EXPOSE 8080
USER 1001

ENTRYPOINT [ "/deployments/run-java.sh" ]
```

docker-compose.yml

```yaml
version: '3.8'
services:
  proxy:
    build: ./proxy
    restart: on-failure
    ports:
      - '80:80'
      - '443:443'
    networks:
      - internal
    depends_on:
      - app

  app:
    build:
      context: ../
      dockerfile: ./containerized/app/Dockerfile
    restart: on-failure
    environment:
      - DB_URL=jdbc:mariadb://db:3306/ssbd06
      - DB_ADMIN_PASSWORD=12345
      - DB_AUTH_PASSWORD=12345
      - DB_MOK_PASSWORD=12345
      - DB_MOL_PASSWORD=12345
    networks:
      - internal
    depends_on:
      - db

  db:
    image: mariadb:11.1
    restart: always
    environment:
      - MARIADB_ROOT_HOST=admin
      - MARIADB_ROOT_PASSWORD=admin
    volumes:
      - db:/var/lib/mysql
      - ./db/docker-init.sql:/docker-entrypoint-initdb.d/docker-init.sql
    networks:
      - internal

volumes:
  db:
    driver: local

networks:
  internal:
    driver: bridge
```

Polecenie lub sekwencję poleceń prowadzących do pełnego uruchomienia aplikacji, wraz ze wszystkimi komunikatami generowanymi w terminalu

```
docker-compose up --scale app=3
```