Ingeniería de Software 3
TRABAJO PRÁCTICO N°6

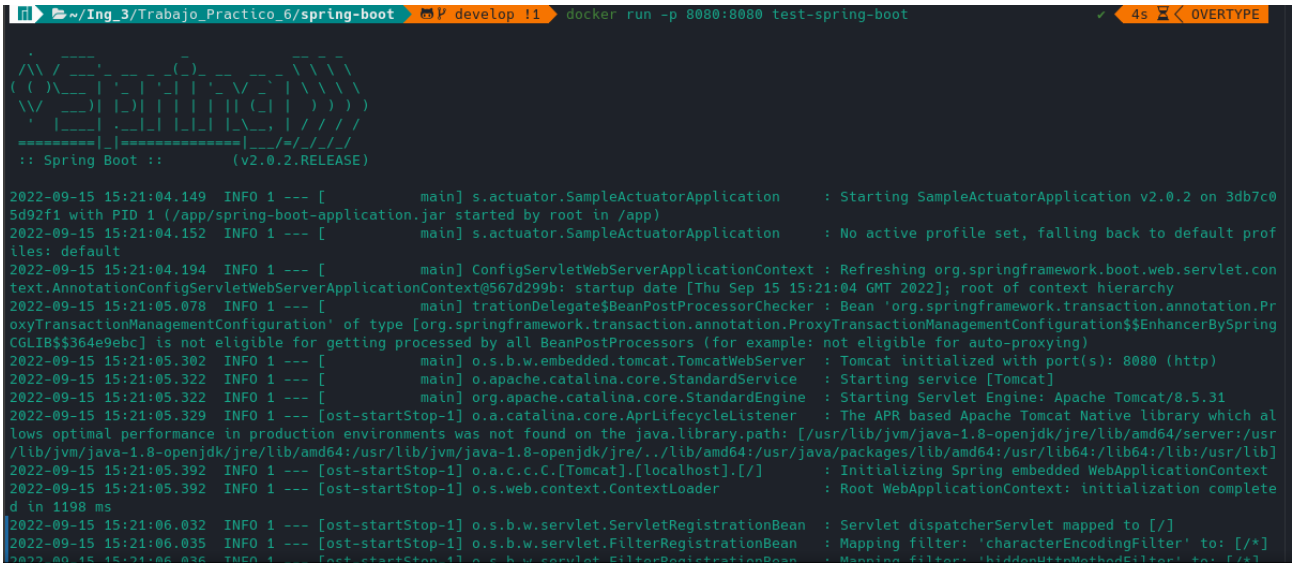
Profesor: Ing Fernando Bono.

Alumno: Menel Angelo (1804789)

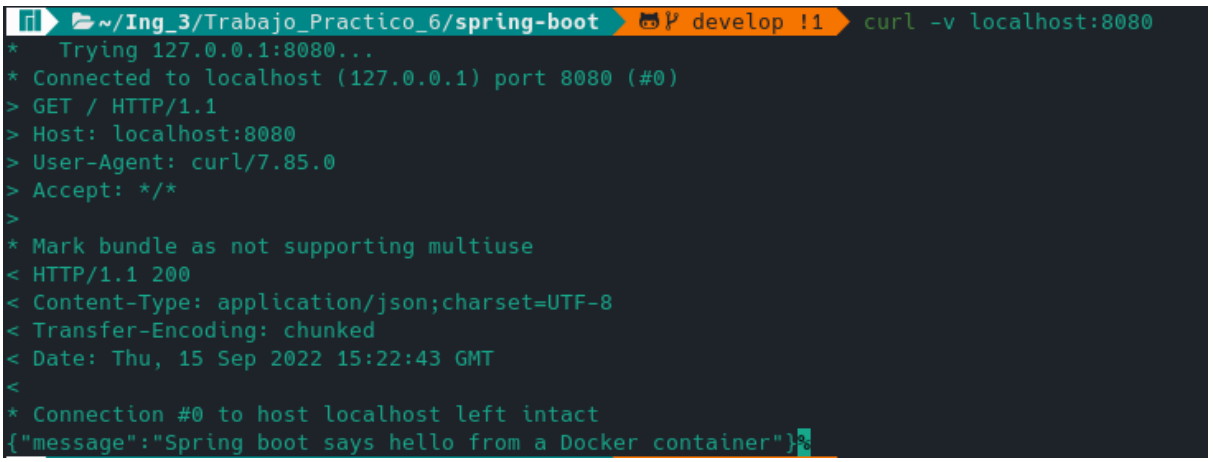
CONCEPTOS DE DOCKERFILE

- FROM: Se le indica el nombre de la imagen a utilizar.
- RUN: Corre un comando de consola.
- ADD: Agrega archivos o directorios a la imagen.
- COPY: Realiza la copia de un directorio a otro.
- EXPOSE: Le indica al contenedor que puerto tiene que estar escuchando.
- CMD: Se especifican los parámetros que utilizará el ENTRYPOINT.
- ENTRYPOINT: Especifica el ejecutable que usará el contenedor.

docker run -p 8080:8080 test-spring-boot



curl -v localhost:8080



Luego agrego al Dockerfile un multistage, con una imagen de maven y otra de openjdk.

```
FROM maven:3.5.2-jdk-8-alpine AS MAVEN_TOOL_CHAIN
COPY pom.xml /tmp/
RUN mvn -B dependency:go-offline -f /tmp/pom.xml -s /usr/share/maven/ref/settings-docker.xml
COPY src /tmp/src/
WORKDIR /tmp/
RUN mvn -B -s /usr/share/maven/ref/settings-docker.xml package

FROM java:8-jre-alpine

EXPOSE 8080

RUN mkdir /app
COPY --from=MAVEN_TOOL_CHAIN /tmp/target/*.jar /app/spring-boot-application.jar

ENV JAVA_OPTS="-Xms32m -Xmx128m"

ENTRYPOINT exec java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app/spring-boot-application.jar

HEALTHCHECK --interval=1m --timeout=3s CMD wget -q -T 3 -s http://localhost:8080/actuator/health/ || exit 1
```

- Lo que estoy haciendo es definir 2 imágenes dentro del Dockerfile, donde uno sera el de maven. Copio el pom.xml al directorio TMP, corro maven, copio el directorio de nuestra app al TMP/SRC, le indico que el directorio de trabajo es TMP.
- Para java, le indico el puerto al que tiene que escuchar, le pido que crea el directorio app, luego en el COPY creo el enlace entre java y maven. El ENTRYPOINT indica el archivo jar a ejecutar.
- HEALTHCHECK es un comando para crear un endpoint que te indica si el contenedor esta levantado o no.

¿Para qué está la key build.context en el docker-compose.yml?

- El build.context se utiliza para poder utilizar la imagen para distintos contextos, como uno de producción y otro de desarrollo.
- El path al directorio que tiene el Dockerfile.

EL DOCKER PUSH DEVUELVE

```
~/Ing_3/Trabajo_Practico_6/nodejs-docker  develop !1  docker tag test-node angemenel98/test-node:latest
~/Ing_3/Trabajo_Practico_6/nodejs-docker  develop !1  docker push angemenel98/test-node:latest
The push refers to repository [docker.io/angemenel98/test-node]
617da07ff09f: Pushed
ffb8a2f8e681: Pushed
1c9b1d61dd08: Pushed
f1ed0bba6314: Mounted from library/node
2808ff9120f2: Mounted from library/node
cb6eda6d73f0: Mounted from library/node
994393dc58e7: Mounted from library/node
latest: digest: sha256:4eb94077eeec2e0f41e8f5f5765920a8748a0803e08d645fbe8bc0d4e4c6c9ce size: 1788
```