**Docker Commands - Creating Image Manually**

* docker run -dit openjdk:8-jdk-alpine
* docker container exec naughty\_knuth ls /tmp
* docker container cp target/hello-world-rest-api.jar naughty\_knuth:/tmp
* docker container exec naughty\_knuth ls /tmp
* docker container commit naughty\_knuth in28min/hello-world-rest-api:manual1
* docker run in28min/hello-world-rest-api:manual1
* docker container ls
* docker container commit --change='CMD ["java","-jar","/tmp/hello-world-rest-api.jar"]' naughty\_knuth in28min/hello-world-rest-api:manual2
* docker run -p 8080:8080 in28min/hello-world-rest-api:manual2

**Quick Tip for Windows 10 : Use 192.168.99.100 in URL instead of localhost**

If you are using **Window 10** and are using **docker toolbox**

=> Use **192.168.99.100** instead of **localhost**.

**Note:**If **192.168.99.100** does not work, you can find the IP by using the command docker-machine ip

docker run -p 5000:5000 in28min/todo-rest-api-h2:1.0.0.RELEASE

**Dockerfile: Basic**

Describes specific instructions to execute in order to build an image

**Instructions**

* FROM openjdk:8-jdk-alpine

(base image that the container will be created in)

* ADD target/hello-world-rest-api.jar hello-world-rest-api.jar

(copies the image)

* ENTRYPOINT ["sh", "-c", "java -jar /hello-world-rest-api.jar"]

(Sets command to run at startup/instruction in the shell)

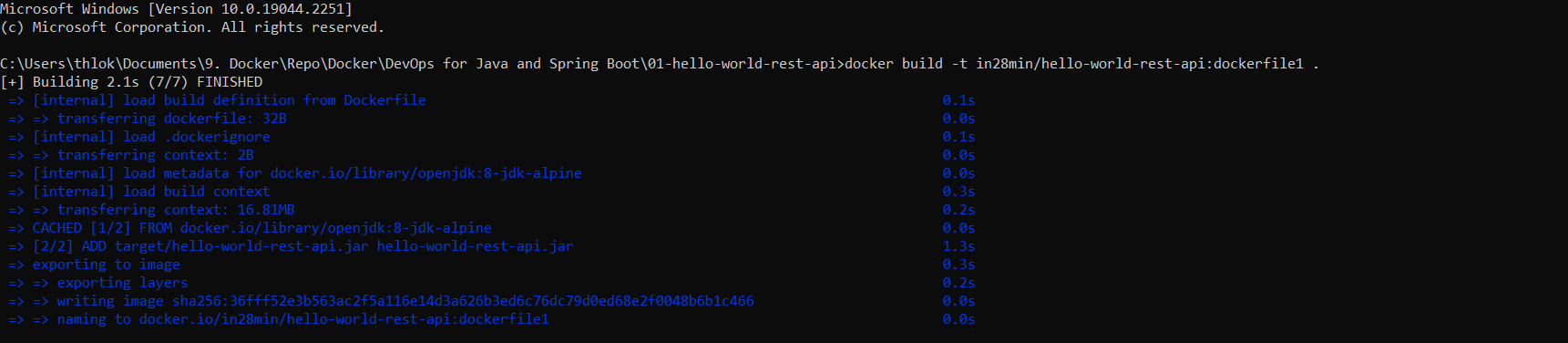
**Docker Commands - Creating Image Aumatically**

**Building an Image from a project in eclipse**

1.Build a Jar - /target/hello-world-rest-api.jar

- run as, maven build, Goals: package DskipTests / from cmd: mvn package -DskipTests

- docker build -t in28min/hello-world-rest-api:dockerfile1 .

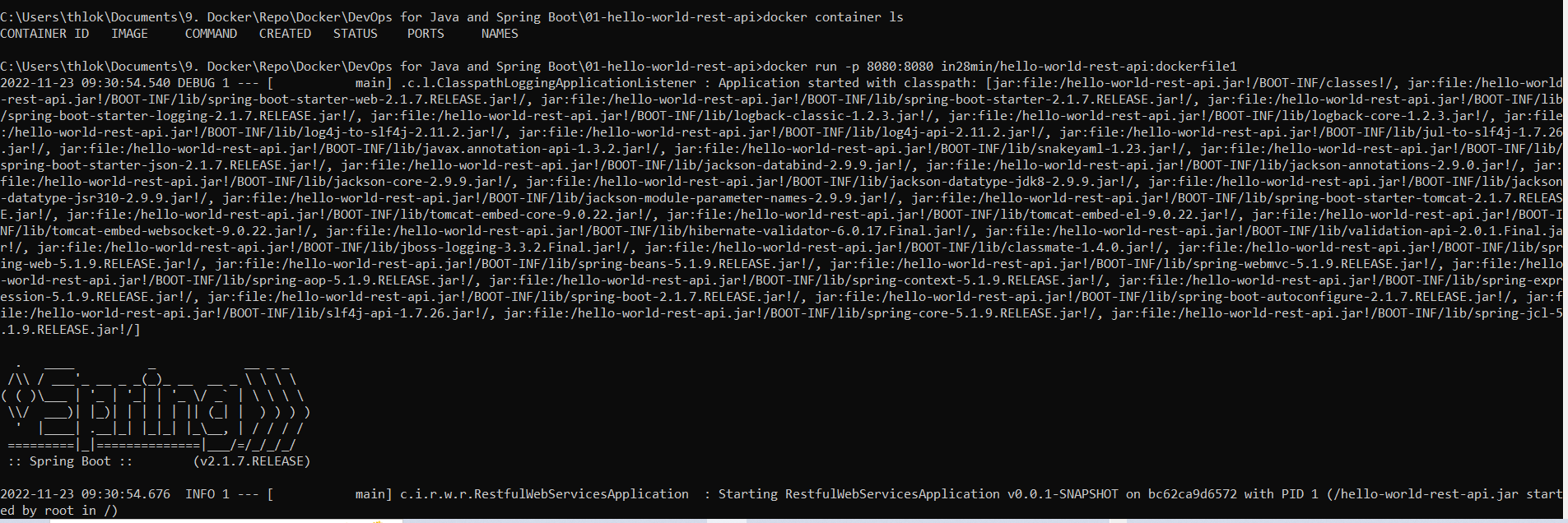


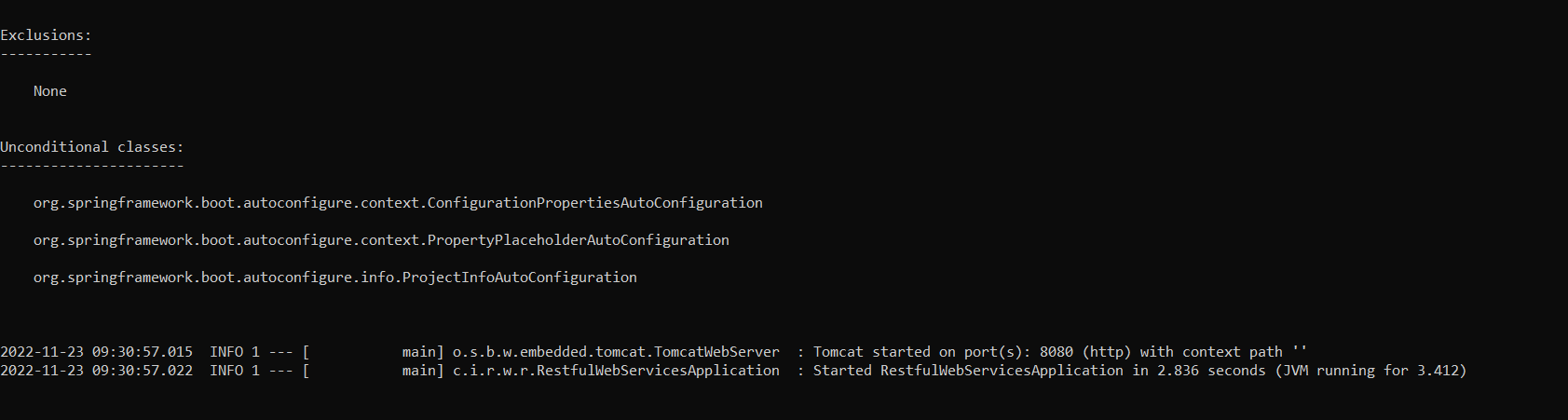
2.Setup the Prerequisites for Running the JAR - openjdk:8-jdk-alpine

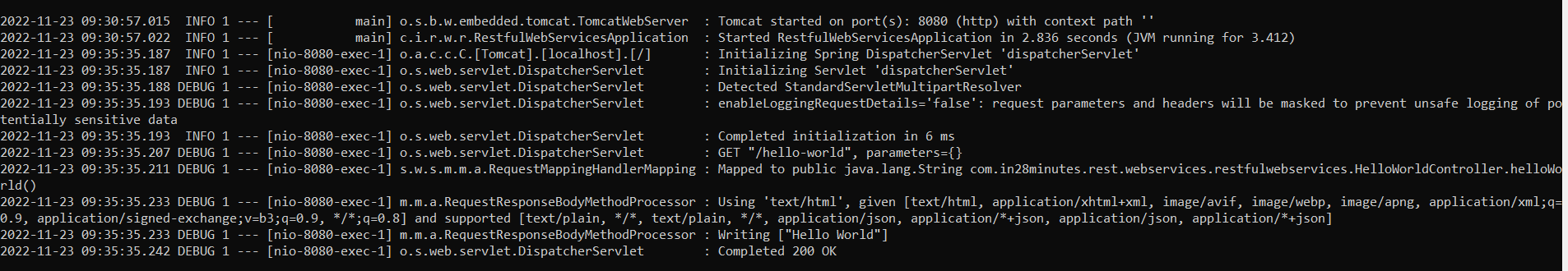
3.Copy the jar

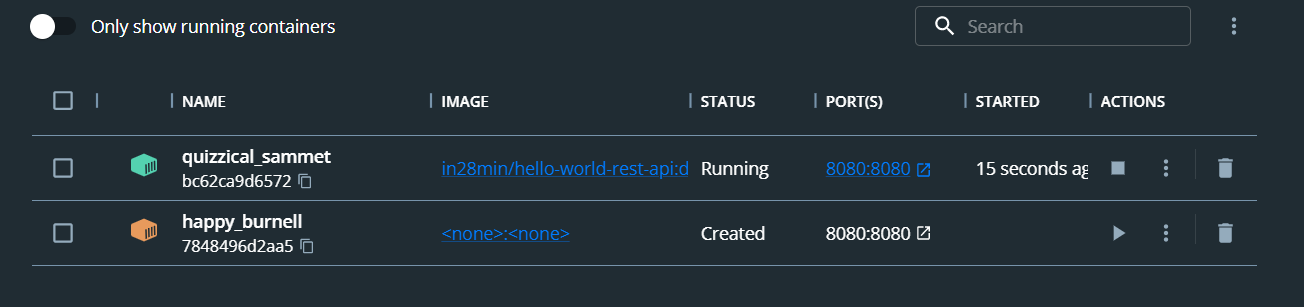
4.Run the jar

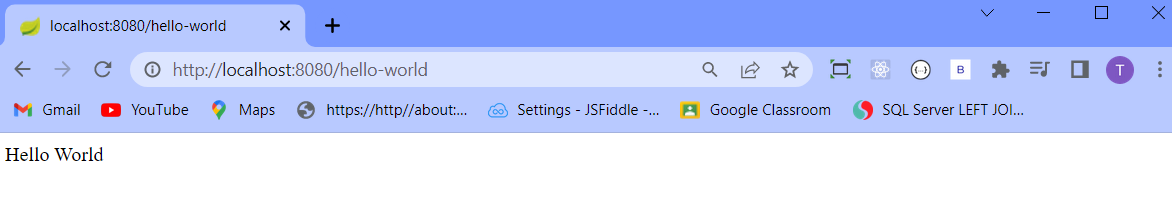
Docker run -p 8080:8080 in28min/hello-world-rest-api:dockerfile1



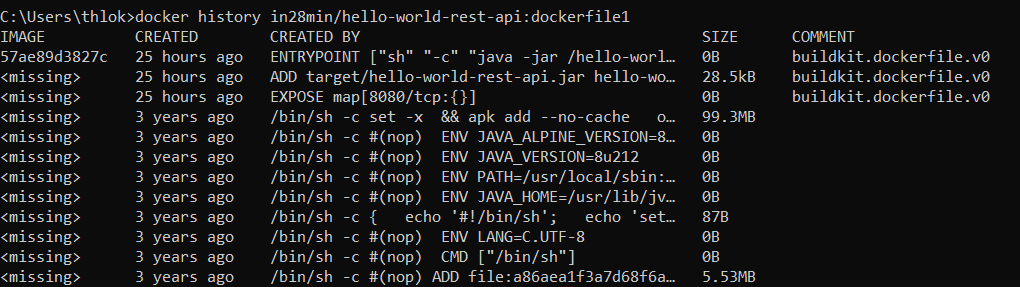








**docker history (gives history of the image)**

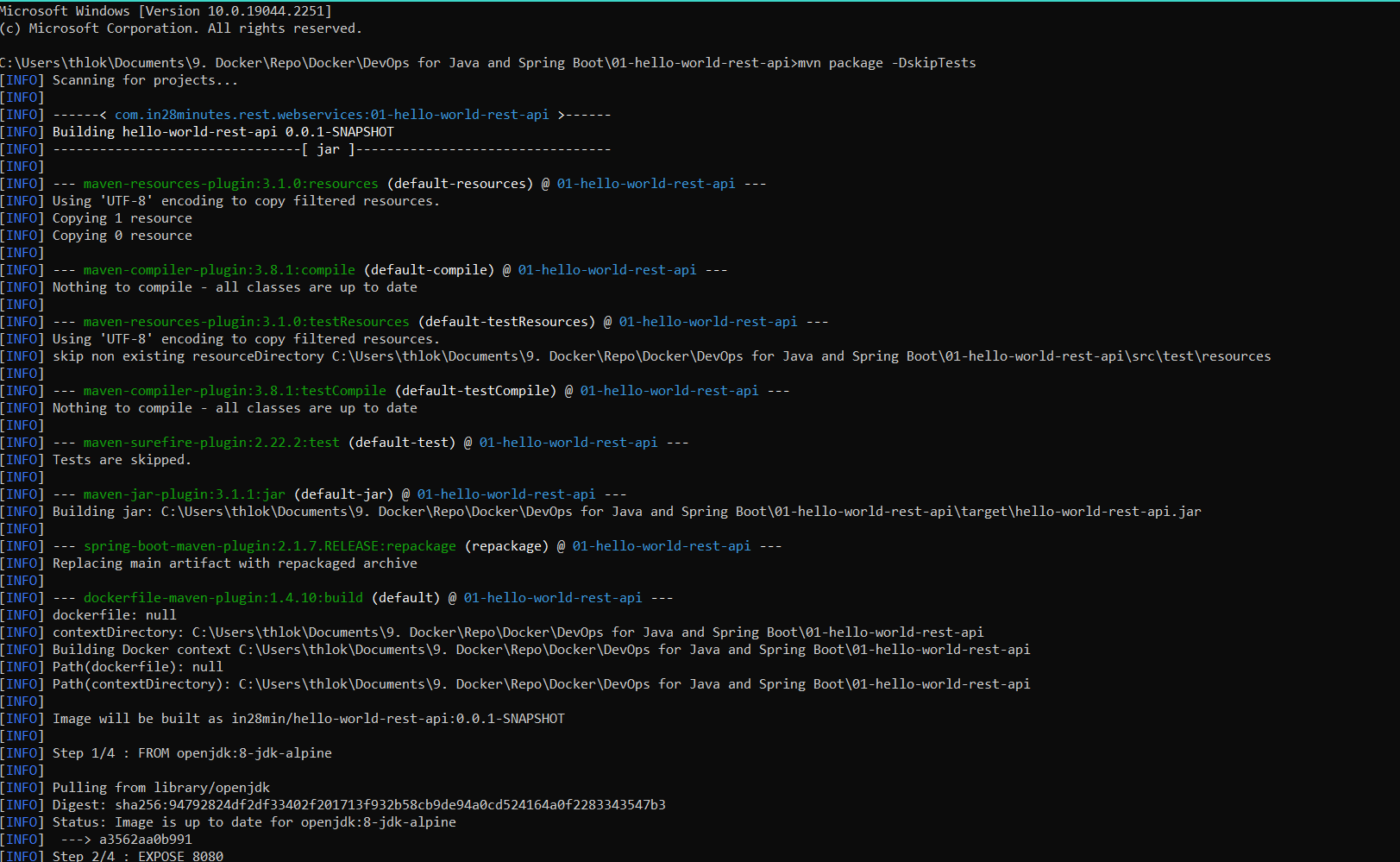


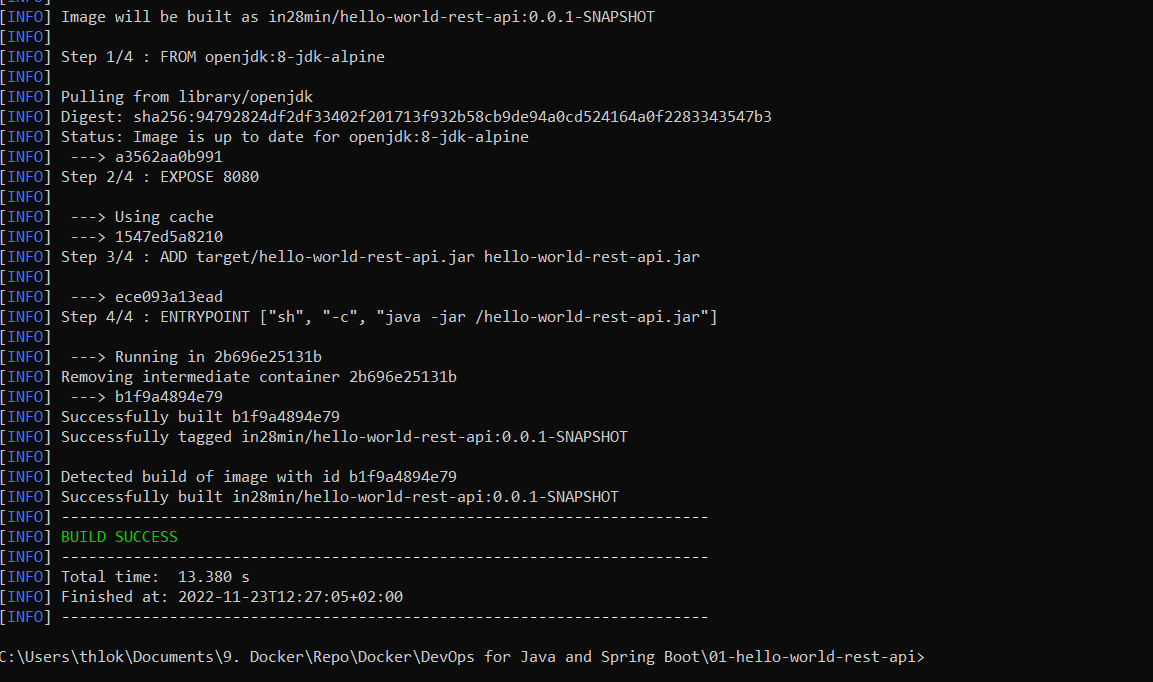
**Building image using maven build in eclipse**

* Maven build >> insert under Goals “package -DskipTests”

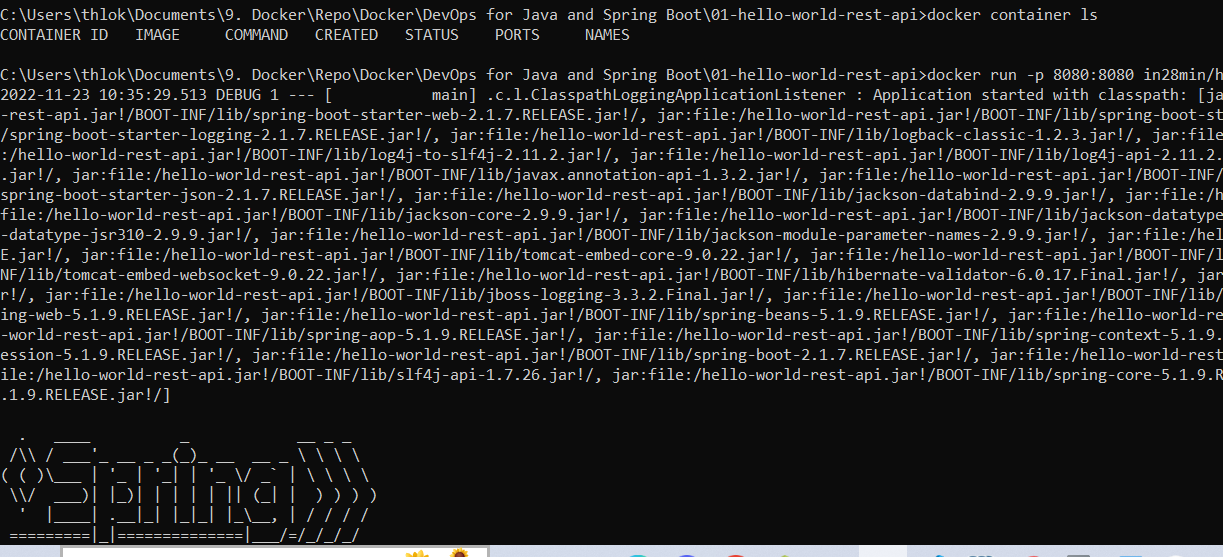
**Building image in cmd**

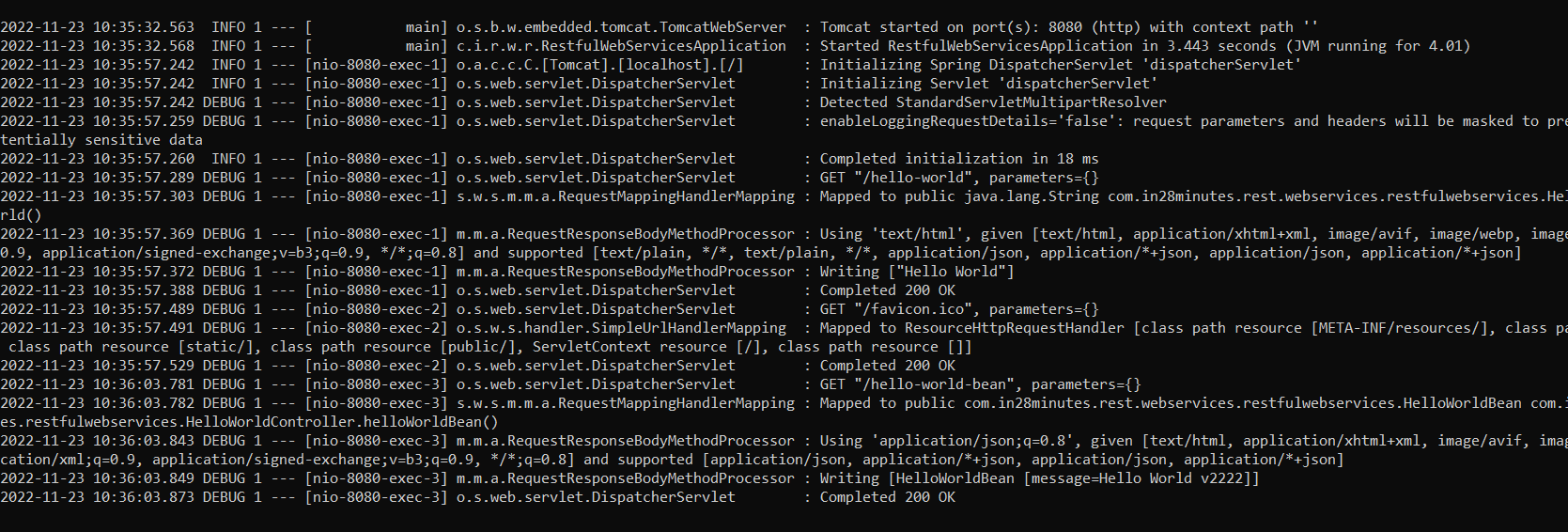
* Mvn package -DskipTests

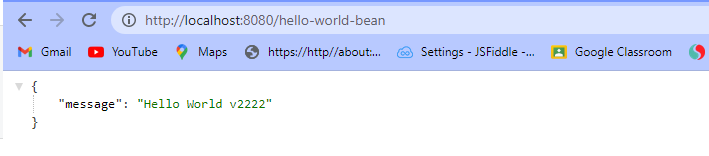


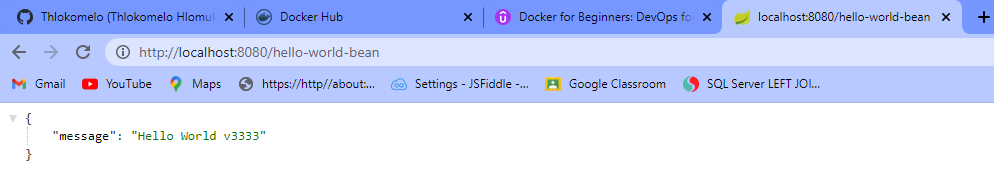


**Running the image**

docker run -p 8080:8080 in28min/hello-world-rest-api:0.0.1-SNAPSHOT







**Other ways to create images**

**Creating generic Dockerfile**

FROM openjdk:8-jdk-alpine

EXPOSE 8080

ADD target/\*.jar app.jar

ENTRYPOINT ["sh", "-c", "java -jar /app.jar"]

**Run this after changing the Dockerfile and adding maven plugin**

* mvn clean package -DskipTests

**Creating Level 2 Dockerfile**

FROM openjdk:8-jdk-alpine

//(creates variable in dependency folder)

ARG DEPENDENCY=target/dependency

COPY ${DEPENDENCY}/BOOT-INF/lib /app/lib

COPY ${DEPENDENCY}/META-INF /app/META-INF

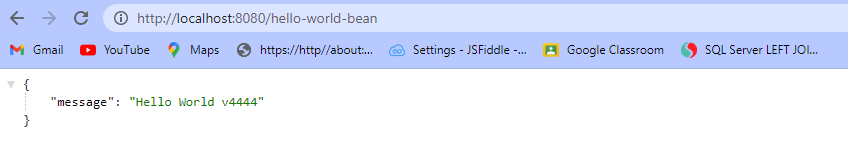
COPY ${DEPENDENCY}/BOOT-INF/classes /app

//runs the webservice using all the liabraries

ENTRYPOINT ["java","-cp","app:app/lib/\*","com.in28minutes.rest.webservices.restfulwebservices.RestfulWebServicesApplication"]

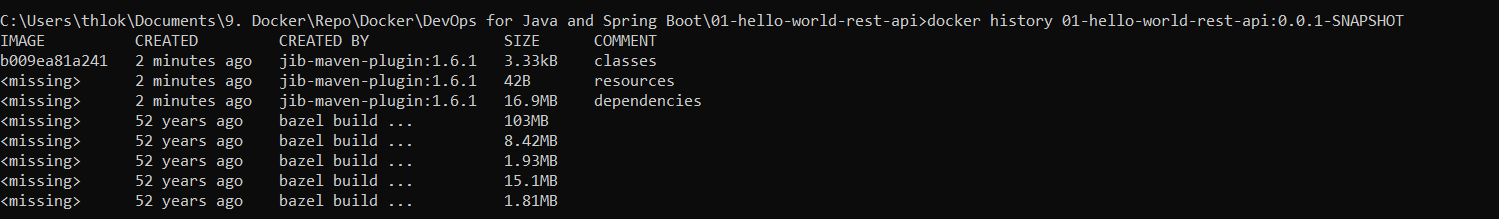
**Build the image**

* Update controller class
* mvn clean package -DskipTests
* docker build -t in28min/hello-rest-api:dockerfile1 .



**JIB Plugin (Java Specific)**

* Simplifies containerization of Java applications without a Dockerfile.
* Automatically detects classes to run the web service.



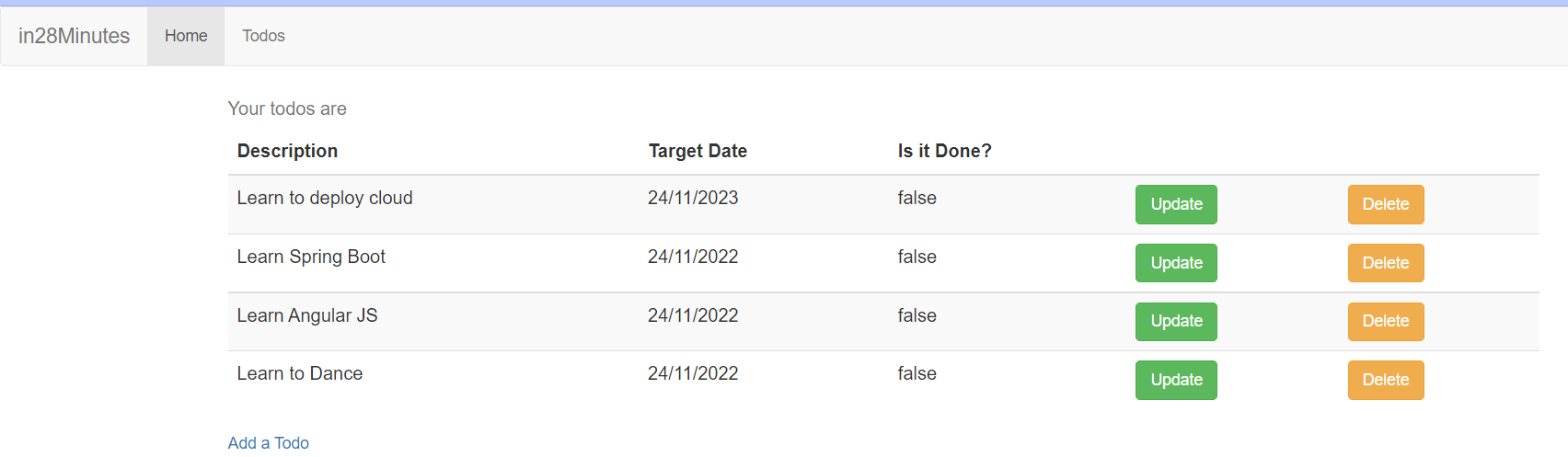
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**02 : To-do we application**

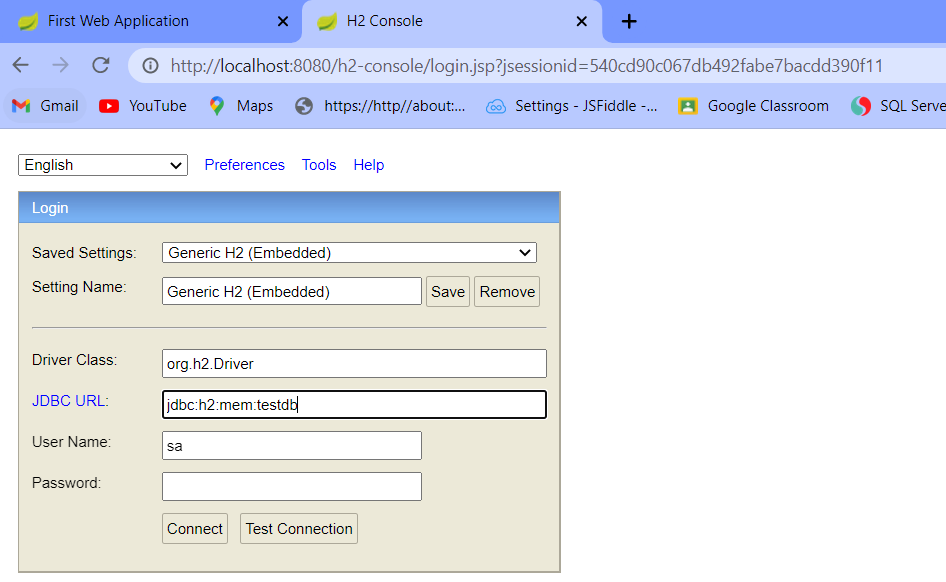
http://localhost:8080/login

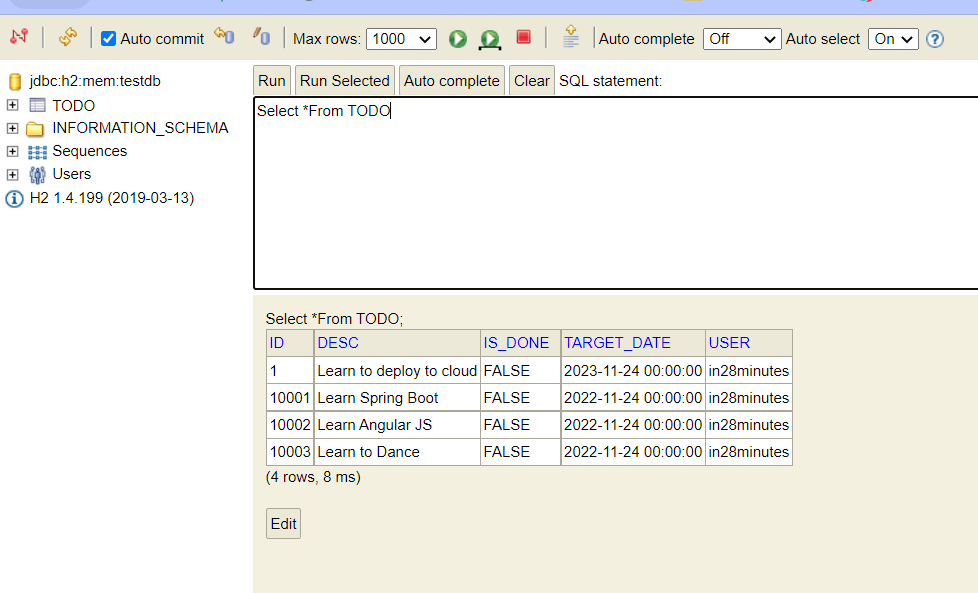
User: in28minutes

Password: dummy



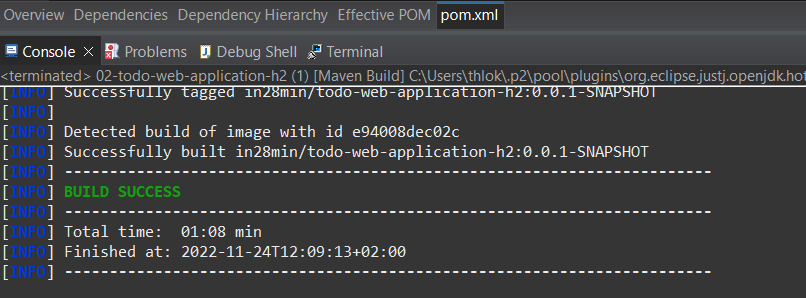
**H2 Console**





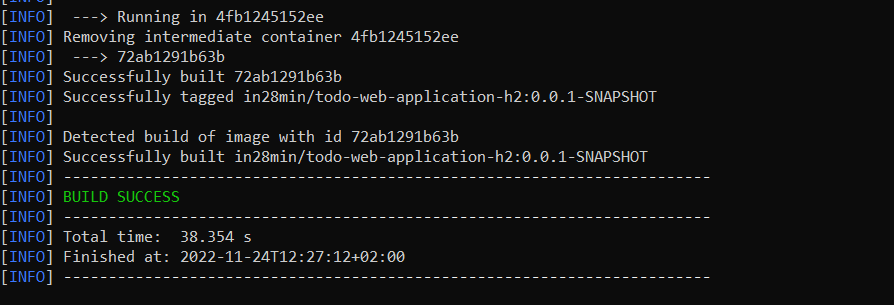
**Deploying application**

* Create a deployable unit and run it from a local machine
  + Run as Maven build: Goals “clean package” (creates a war package and build an image)



**OR**

* + CMD: mvn clean package
  + Uses tomcat to build image instead of jdk



* Run the app
  + docker run -p 8080:8080 in28min/todo-web-application-h2:0.0.1-SNAPSHOT

Dockerfile

FROM tomcat:8.0.51-jre8-alpine

EXPOSE 8080

RUN rm -rf /usr/local/tomcat/webapps/\*

COPY target/\*.war /usr/local/tomcat/webapps/ROOT.war

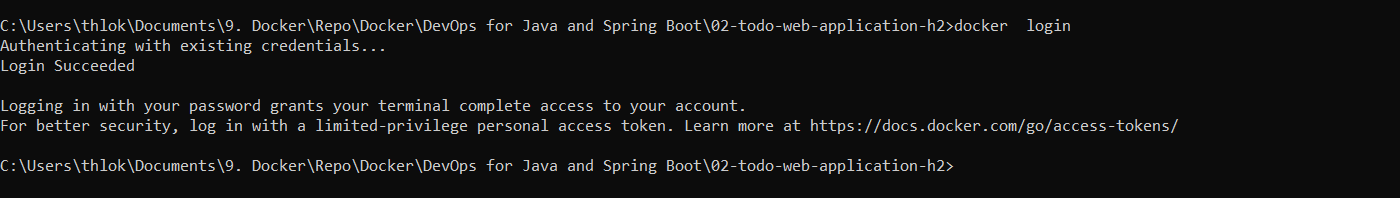
//copies sinple file or directory

CMD ["catalina.sh","run"]

//allows for overriding of original arguments in instruction

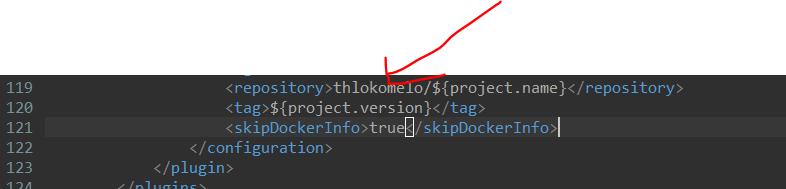
**Pushing image to Docker Hub**

login

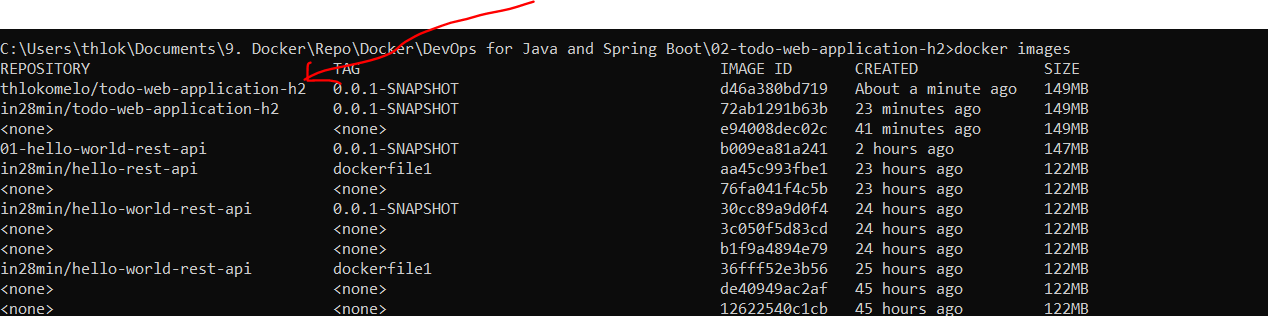


Update pormfile

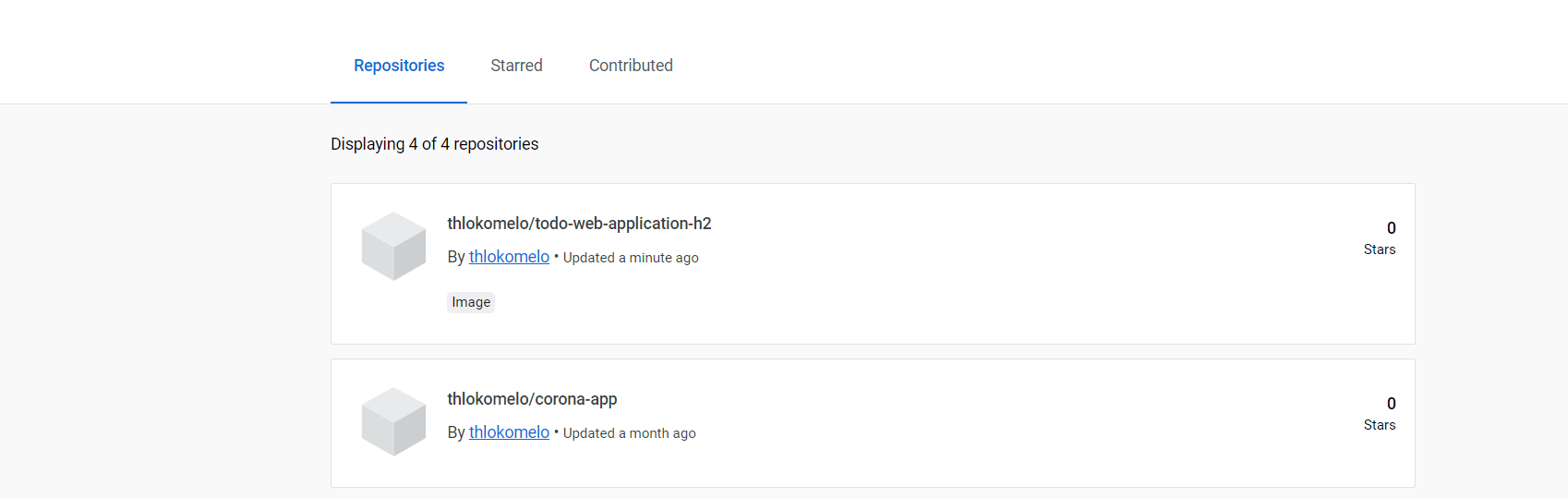
* Ensure username is your own so that docker can push to the right repository



* Run mvn clean package
* Run docker images to see all built images



* Run docker push \*username\* /todo-web-application-h2:0.0.1-SNAPSHOT



--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

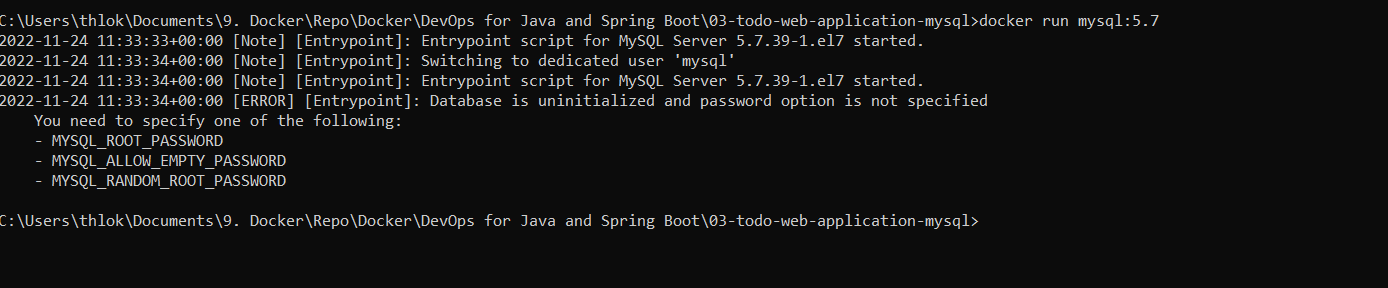
**Connecting and running MYSQL as a Docker container with Java SpringBoot**

**Installing Docker MySQL image**

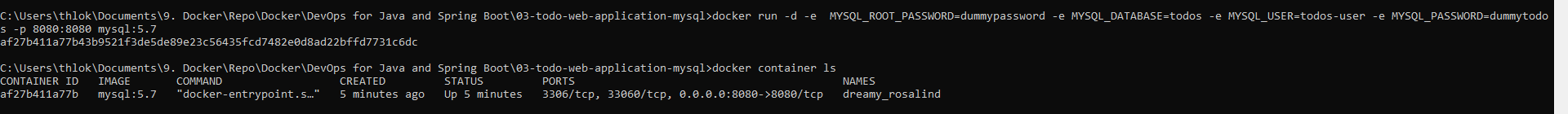
* <https://hub.docker.com/_/mysql>
* Supported tags and respective Dockerfile links
  + 5.7

**Run MySQL image**

* Docker run mysql:5.7 (gives an error)



* docker run -d -e MYSQL\_ROOT\_PASSWORD=dummypassword -e MYSQL\_DATABASE=todos -e MYSQL\_USER=todos-user -e MYSQL\_PASSWORD=dummytodos -p 3306:3306 mysql:5.7



* Login into MySQL
  + \connect
  + \connect todos-user@localhost:3306
  + \Enter password

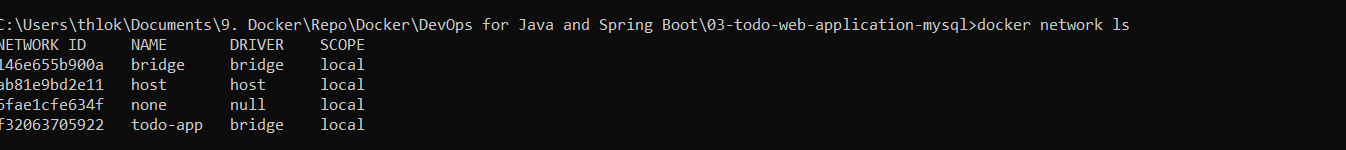


**Create Docker Image**

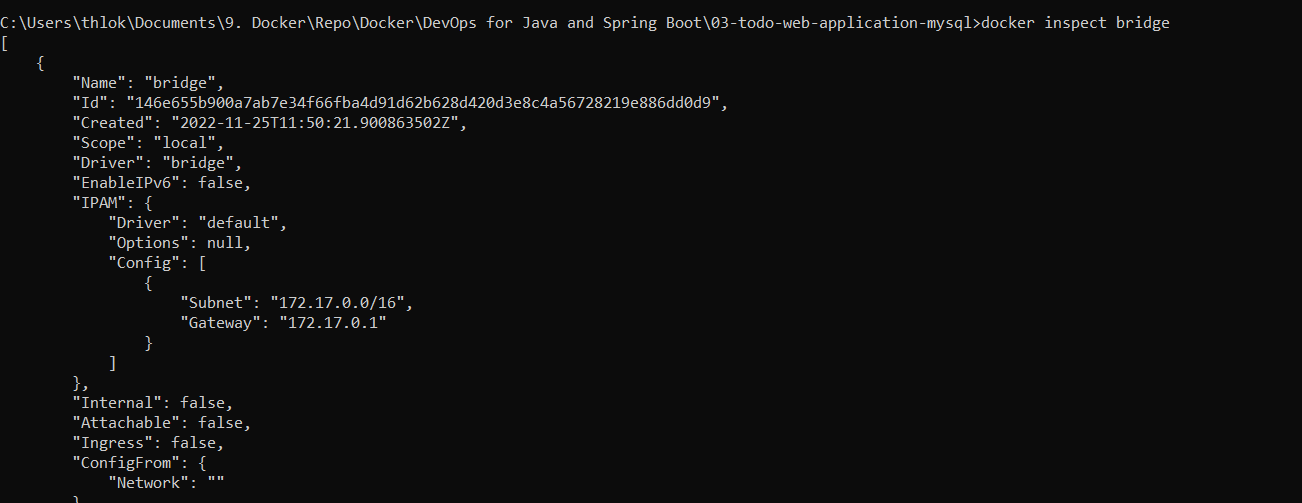
* **mvn clean package -DskipTests**
* **docker container run -p 8080:8080 --link=mysql -e RDS\_HOSTNAME=mysql in28min/todo-web-application-mysql:0.0.1-SNAPSHOT**

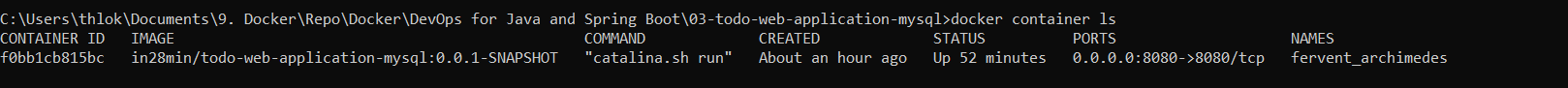
**Viewing all docker networks for Launching options**

* Run docker network ls



* Run docker inspect bridge (To see detailed information on the specific network)



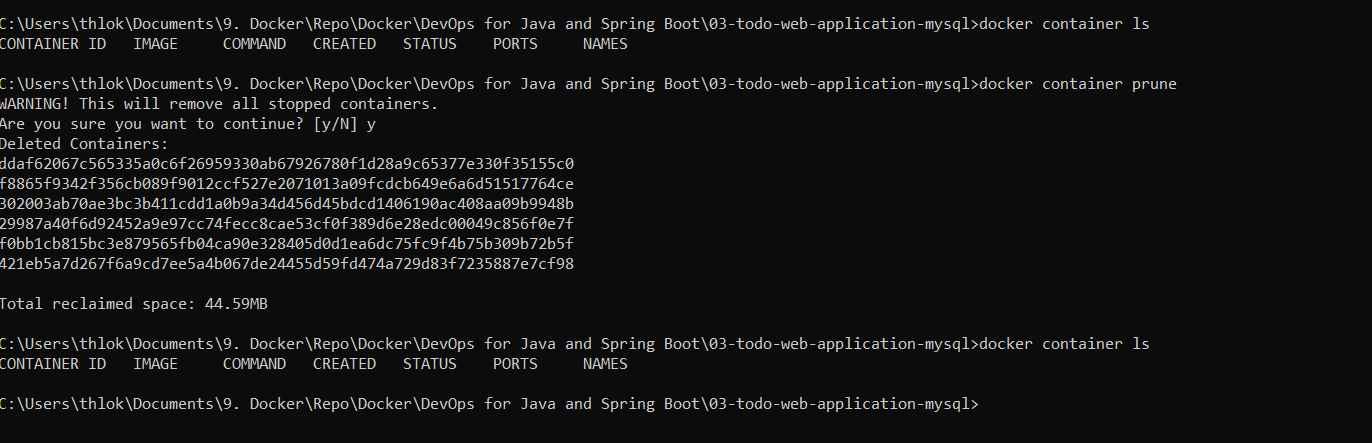


**Launching the app**

* Run docker container run -p 8080:8080 --network=web-application-mysql-network -e RDS\_HOSTNAME=mysql in28min/todo-web-application-mysql:0.0.1-SNAPSHOT

**Docker Volumes**

* **Remove all stopped containers**

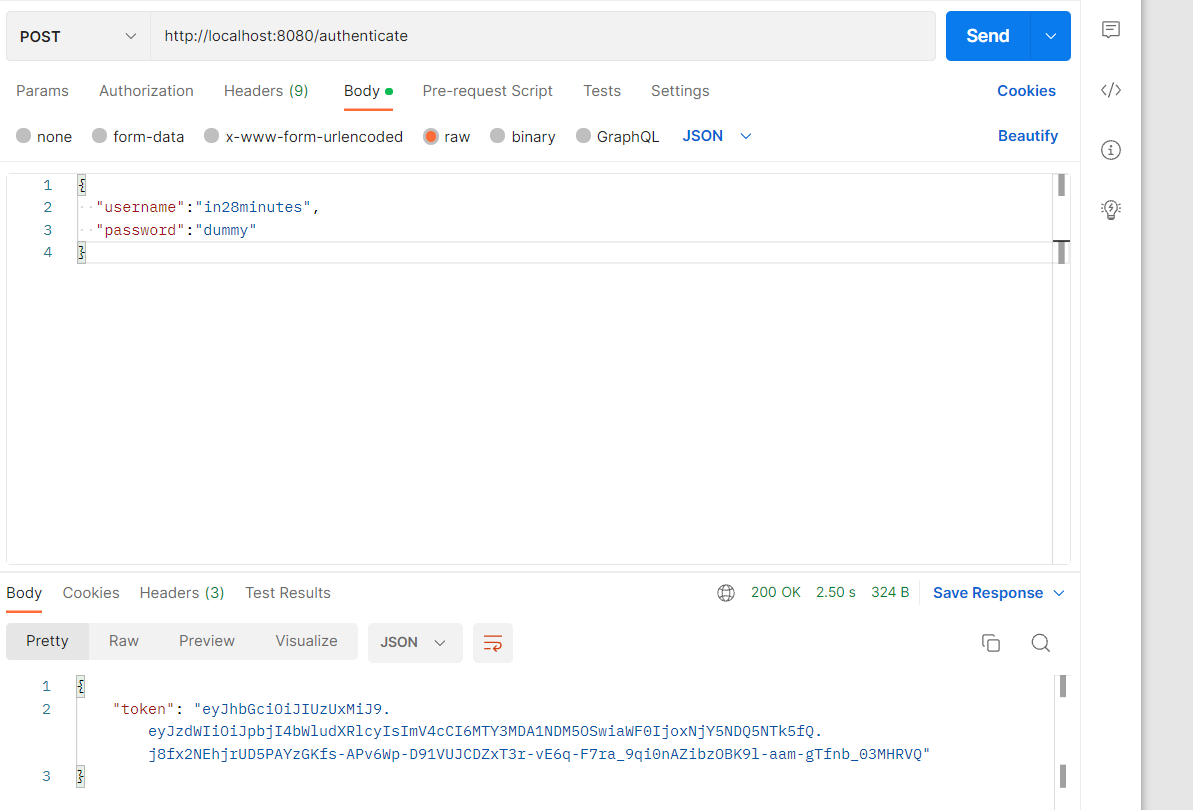


**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

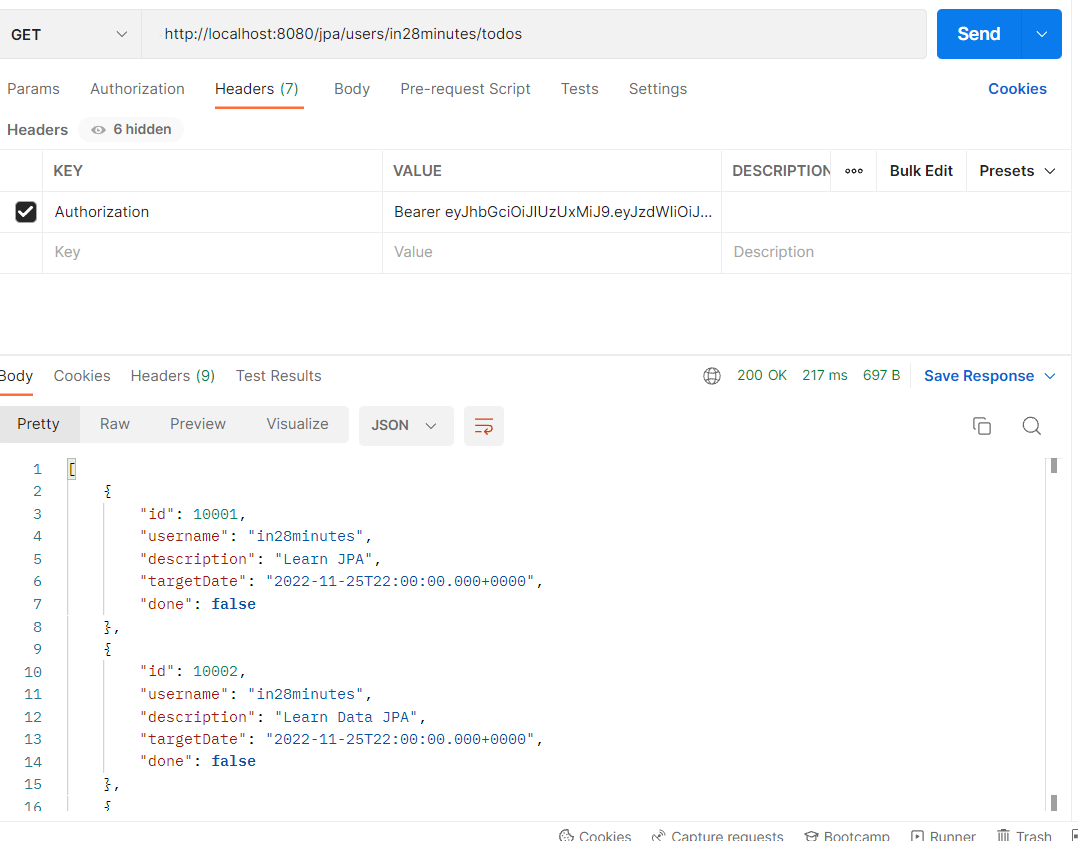
**04-rest-api-full-stack**

**Exploring React backend in Local**

* **Run as java application**
* **Do a post request to authenticate user**

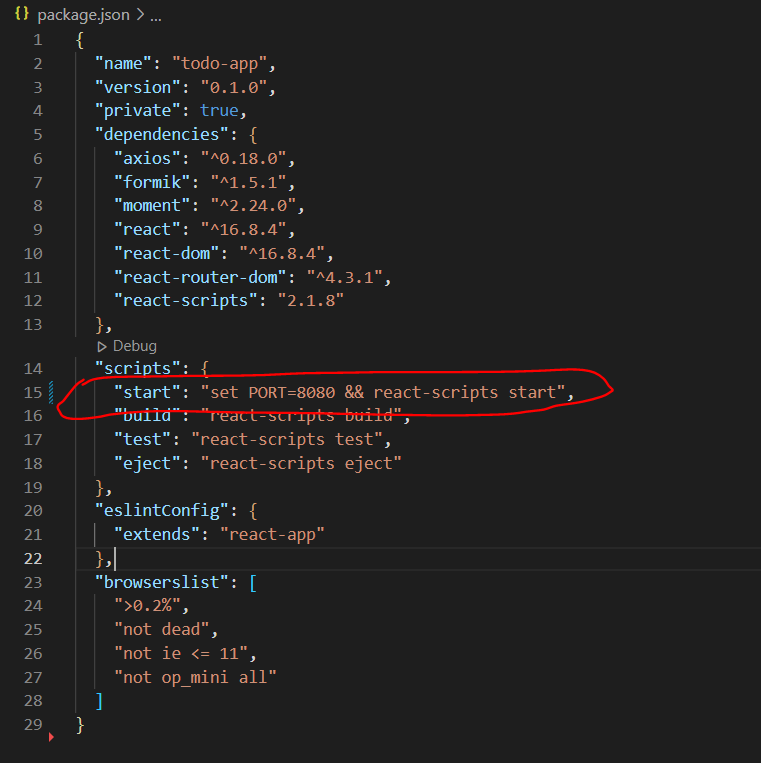


* **Do a get request**

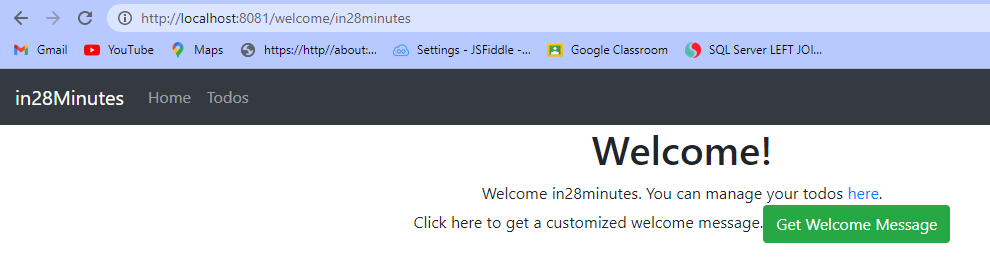


**Run React Frontend in Local**

* **Open Todo app in VSC**
* **Install dependencies**
  + **Npm install**
* **Npm start**
  + - **Change package.js start script**

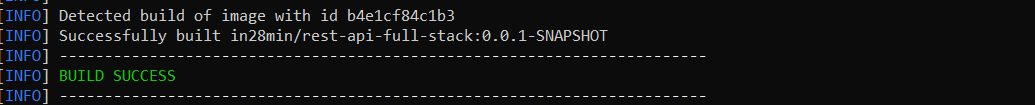


* **Start app on server**

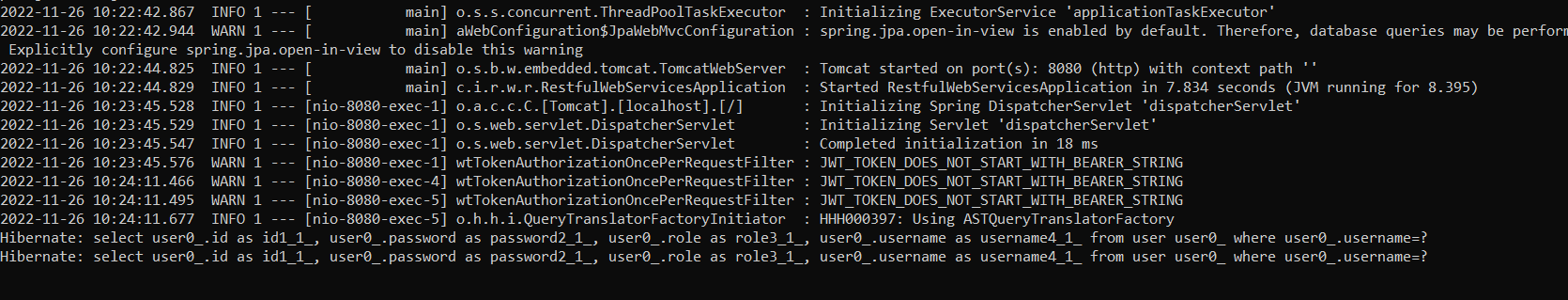


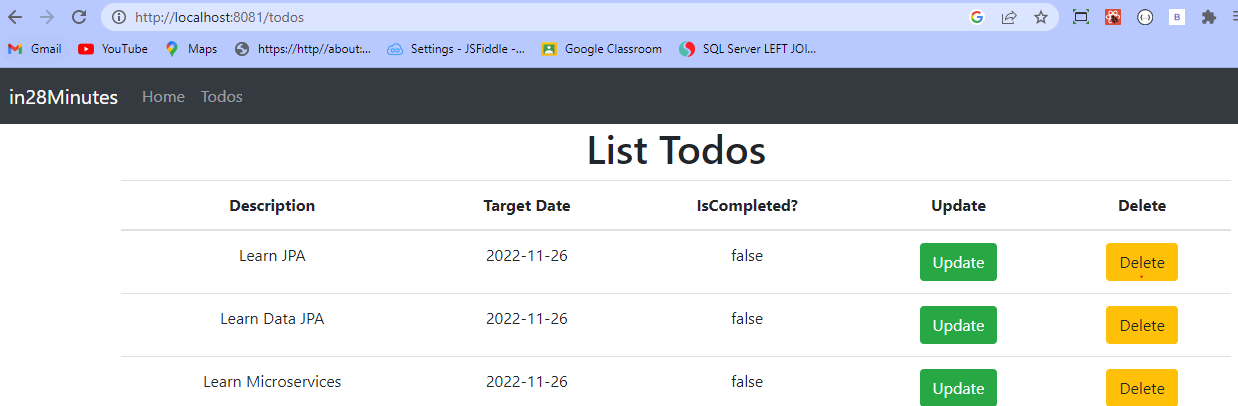
**Containerizing Java Rest API Backend**

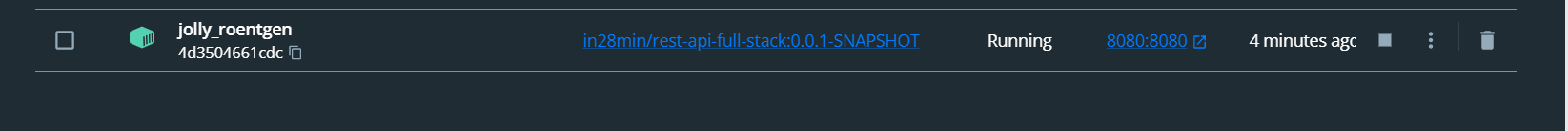
* **Cd to 04-spring-boot-react-full-stack-h2\restful-web-services>**
* **Run mvn clean package to build image**



* **Run docker run -p 8080:8080 in28min/rest-api-full-stack:0.0.1-SNAPSHOT**

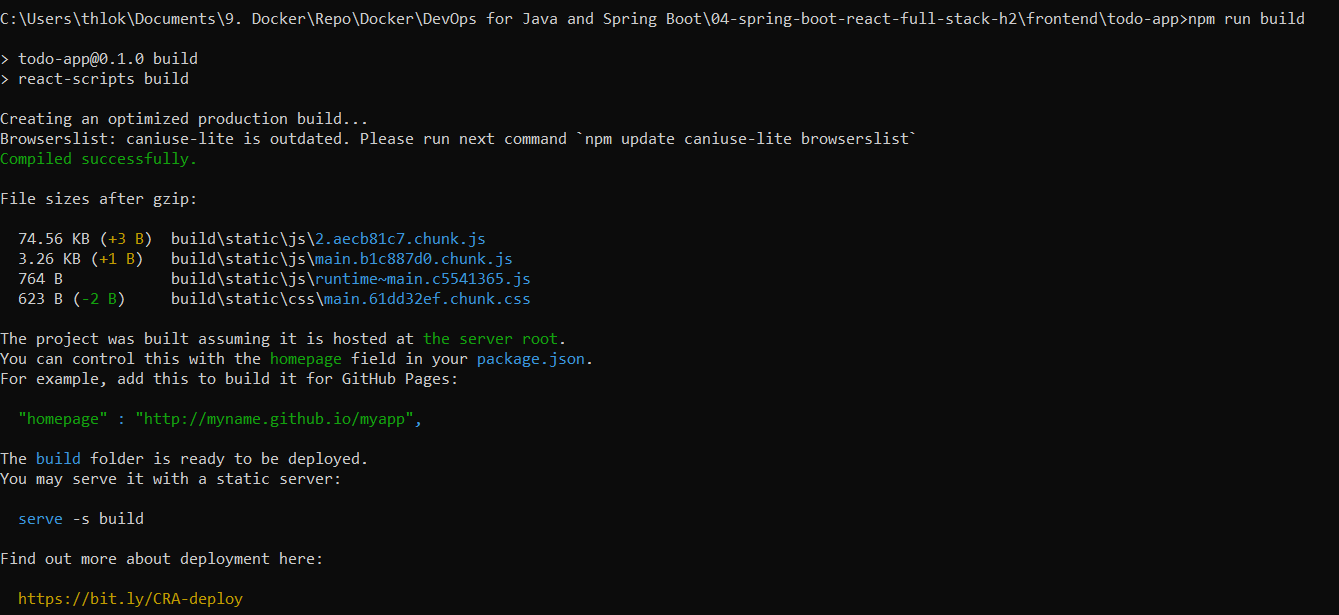






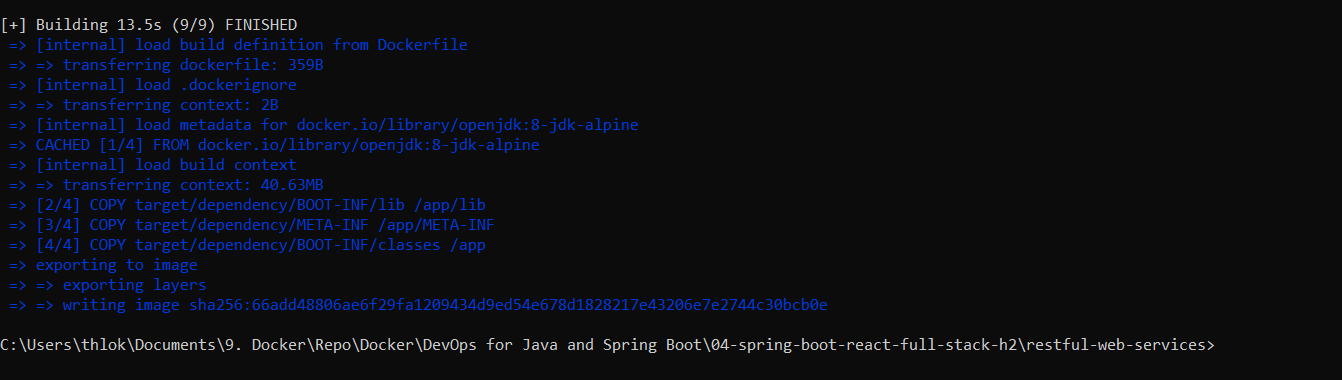
**Creating multi stage docker build for react frontend code**

* **Cd 04-spring-boot-react-full-stack-h2\frontend\todo-app>**
* **Run npm build (reduces project to a small package and can be run from any platform)**

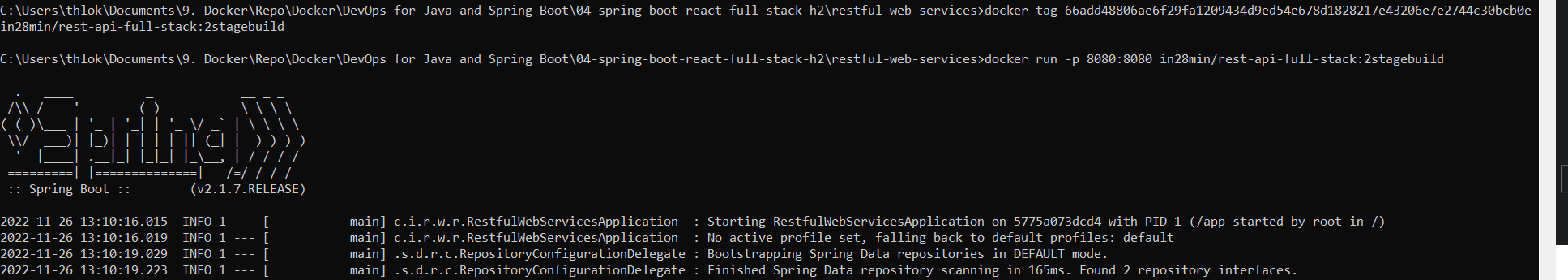


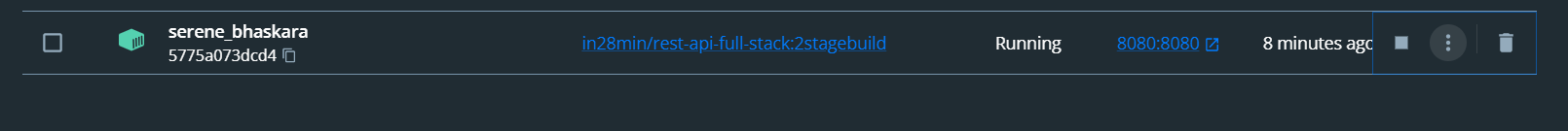
**Create image for nginx**

* Run docker build .



* **Tag and run the image**
  + docker tag 66add48806ae6f29fa1209434d9ed54e678d1828217e43206e7e2744c30bcb0e in28min/rest-api-full-stack:2stagebuild
  + docker run -p 8080:8080 in28min/rest-api-full-stack:2stagebuild

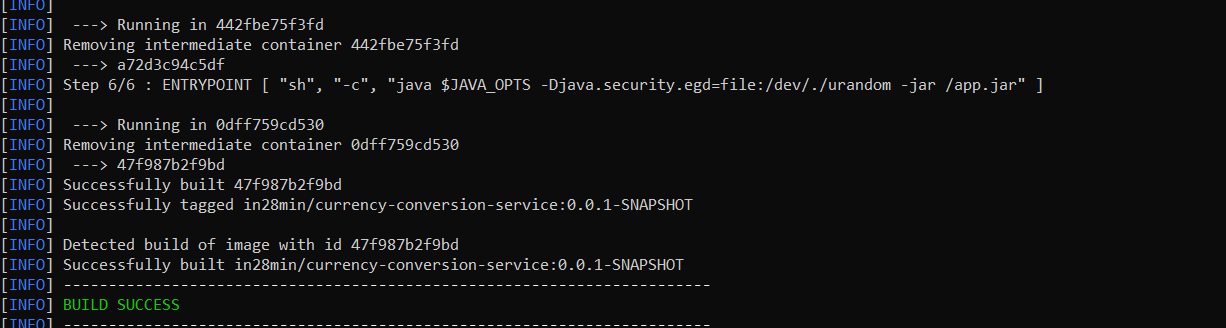




**---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Run Java Spring Boot Microservices**

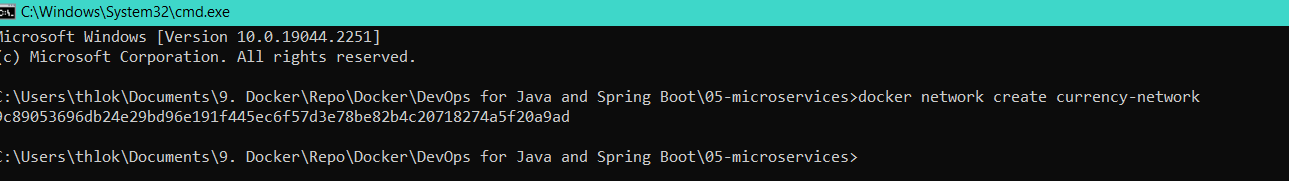
* Cd into specific folders
  + \05-microservices\currency-conversion-service>



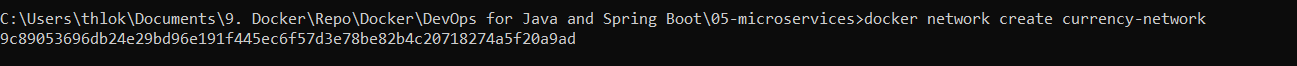
* + \05-microservices\currency-exchange-service>



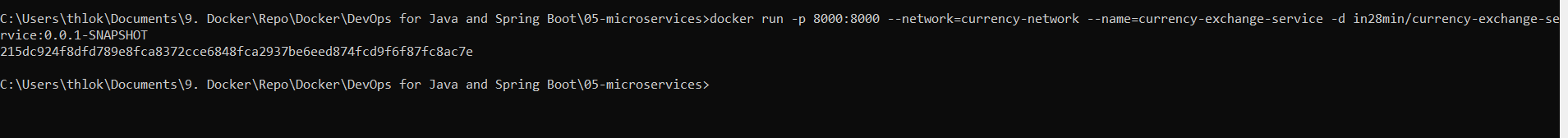
* Run mvn clean package to build images on both projects
* Cd 05-microservices>
  + Run docker network create currency-network

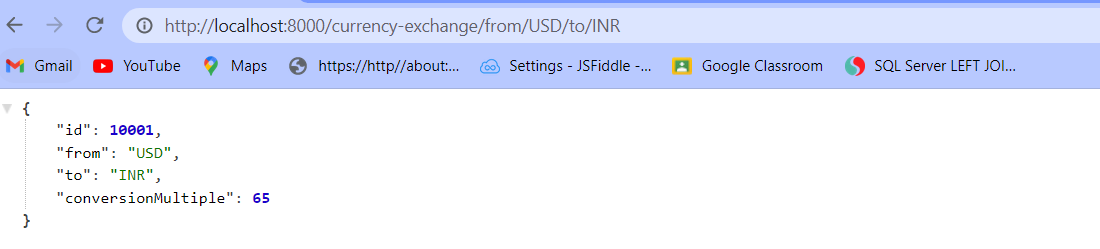


* + Docker network create currency-network

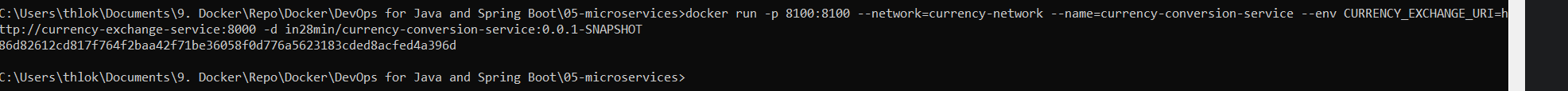


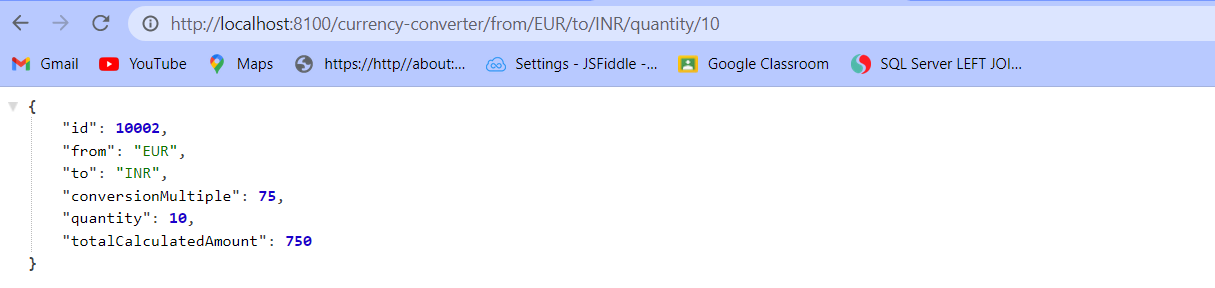
* + docker run --publish 8000:8000 --network currency-network --name=currency-exchange-service in28min/currency-exchange-service:0.0.1-SNAPSHOT





* **docker run -p 8100:8100 --network=currency-network --name=currency-conversion-service --env CURRENCY\_EXCHANGE\_URI=http://currency-exchange-service:8000 -d in28min/currency-conversion-service:0.0.1-SNAPSHOT**





**Docker compose**

* **Cd 05-microservices>**
* **Stop all running apps**
* **Docker run docker-compose up**