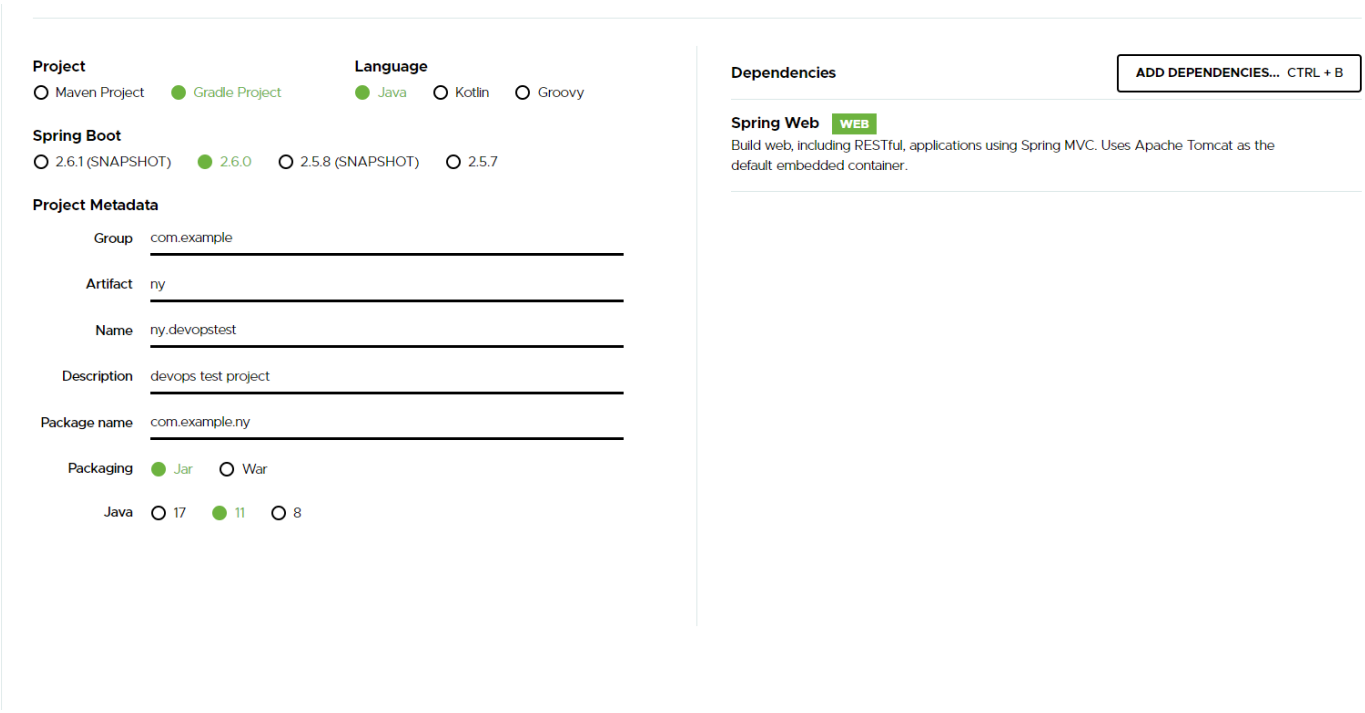
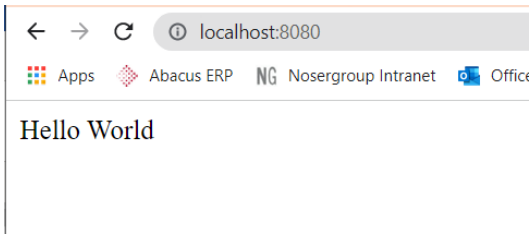
Docker CI/CD

1. Preparation

- Create a DockerHub account
- Create a Heroku Account
    - Create an empty app.
- Setup a simple working spring web project in a new github repo using only the Spring Web dependency. It should serve a single endpoint with a simple message:





2. Build a docker image using this dockerFile

```
FROM gradle:jdk11 AS build
COPY --chown=gradle:gradle . /home/gradle/src
WORKDIR /home/gradle/src
RUN gradle --no-daemon bootJar

FROM openjdk:11
RUN mkdir /app
COPY --from=build /home/gradle/src/build/libs/*.jar /app/spring-boot-application.jar
EXPOSE 8080
CMD ["java","-jar", "-Xmx4g", "/app/spring-boot-application.jar"]
```

To Test your Solution, try to run your image from the command line, so that you can see the page in your browser

3. Create A GithubAction that Builds the Docker Image each time you push to the main branch:

4. Modify the Action such that the image is instead pushed to your Dockerhub Repo.

HINT: For this you will need to provide your Docker Credentials as Secrets to the Action

To check your solution, try to pull the image from the repo and run it locally. Make sure its the remote version, not a local instance.

HINT: for this you may need to execute the «docker login» locally first

5. Modify your app and the Image such that you can pass the Port your app runs on as an environment variable when you start up a container.

6. Modify the Action such that in addition, the image is deployed to your HEROKU App.

CONGRATULATIONS: you just deployed your first app!

7. Modify your Github Action to run a Unit test before pushing your container to Heroku (only if the Test passes)

8. Extra : Try to get your üK-Application to run on Heroku.

   For this you will need to swap the PostgreSQL database for an in-memory database (H2), as PostgreSQL support is a paid addon in Heroku. Thanks to Spring its very easy to do that, just replace the PostgreSQL dependency in build.gradle with this line :

```
runtimeOnly 'com.h2database:h2'
```