Building your first container

Learning Outcomes

After completing the lab, you will be able to:

1. Describe how to create Docker container
2. Describe how to write a docker file
3. Run your Spring boot application as a docker container

Dockerizing

1. Create a new file named `Dockerfile` inside root project folder
2. Add instructions to download the base image. In order to run Java application using JDK 11, use - adoptopenjdk:11-jre-openj9
3. Add instructions to copy the dependencies & build artifacts(jar/war) from the local directory into the docker image
4. Provide a command or an entrypoint to start the application within the docker container

   Example snippet for reference

   ```
   FROM adoptopenjdk:11-jre-openj9
   ARG JAR_FILE=build/libs/pages.jar
   COPY ${JAR_FILE} app.jar
   ENTRYPOINT ["java","-jar","/app.jar"]
   ```

5. Building the Docker Image

   The next step is to build the docker image. To ensure that the jar file is available locally build the project once using the gradle command.

   ```
   ./gradlew clean build
   ```

6. Example snippet to build the docker image

   ```
   docker build -t pages .
   ```

If you get an error containing `Permission Denied while trying to connect to the docker daemon`

`socket` you will need to execute the command `sudo chmod 666 /var/run/docker.sock`

7. After it finishes, run this to see the image it has built

```
docker image list
```

8. Run the image as a container

```
docker run -p 8080:8080 pages
```

In the run command, we have specified that the port 8080 on the container should be mapped to the port 8080 on the Host OS.

Once the application is started, you should be able to access it at http://localhost:8080

The container runs in the foreground. You can run the container in the background using -d option.

Pressing `CTRL + C` sometimes might not stop the process. You will need to manually terminate the container.

   a. Use `docker ps` and fetch the container id.

   b. `docker kill <container-id>`

Pushing the docker image to docker hub

1. Login with your Docker ID to push or pull images from Docker Hub.

   If you don't have a Docker ID, head over to docker hub to create one, before proceeding futher.

```
docker login
```

2. Tag the image using the notation `docker-username/repository:tag.`

```
docker tag pages <docker-username>/pages:1.0
```

   Make sure to replace username with your docker id in the above command.

3. Verify the newly created tagged image

```
docker images
```

4. Push the image to docker hub

```
docker push <docker-username>/pages:1.0
```

5. Pull the image from docker hub and test it on local machine. Kill the container after you test it.

```
docker run -p 8080:8080 <docker-username>/pages:1.0

docker ps

docker kill <container-id>
```

6. Commit your code to your github repository

```
git add .
git commit -m "commit message"
git push -u origin master
```

Troubleshooting guide

1. `docker container list --all` . This will show both running and stopped containers. Note the `CONTAINER-ID` and/or the `NAMES` of the failed container. We'll need it next.
2. `docker container logs CONTAINER-ID` . This shows the console output from the failed container.
3. To know more about docker commands Run `docker -h`

Resource clean up

1. `docker container list` - View running containers. Note the `CONTAINER ID` and/or the `NAMES` of the running container.
2. `docker container stop CONTAINER-ID` .This stops the container.
3. `docker container list` - Verify that the container is now stopped.
4. `docker container rm CONTAINER-ID` - Delete the container.
5. `docker image list` . The image is still there, only the container we created is removed.
6. Verify that the docker image exists with the right tag in docker hub