

Devops tp1

Database :

Basics

`docker build -t margot/tp1_database .`

Why do we need a volume to be attached to our postgres container?

→ to save the data into.

1-1) [Document your database container essentials: commands and Dockerfile.](#)

- Init database

`docker network create app-network`

`docker run -d --network app-network -p 8080:8080 adminer`

`docker run --network app-network --name database -e POSTGRES_DB=db -e POSTGRES_USER=usr -e POSTGRES_PASSWORD=pwd -d postgres`

- Persist data

`docker run --network app-network --name database -e POSTGRES_DB=db -e POSTGRES_USER=usr -e POSTGRES_PASSWORD=pwd -v /home/tp/Desktop/tp1:/var/lib/postgresql/data -d postgres`

Backend API :

Basics:

Dockerfile

```
FROM openjdk:11
COPY Main.java /usr/src/app/
CMD ["java", "/usr/src/app/Main.java"]
```

`docker image build -t docker-java-jar:latest .`

`docker run docker-java-jar:latest`

Multistage build

- Backend simple api

1-2 Why do we need a multistage build? And explain each step of this dockerfile.

To reduce size of a project

step 1 : add controller folder and Greetingcontroller file to the project.

step 2 : add a Dockerfile into the source project at the beginning.

step 3 : move into folder simpleapi (cd)and build docker image

docker image build -t simple-api .

step 4 : run the docker

docker run -d --name java-container -p 8081:8080 simple-api

step 5 : open navigator and watch if it's works

<http://localhost:8081/>

- Backend API

application.yml

```
spring:
  jpa:
    properties:
      hibernate:
        jdbc:
          lob:
            non_contextual_creation: true
            generate-ddl: false
            open-in-view: true
    datasource:
      url: jdbc:postgresql://database:5432/db
      username: usr
      password: pwd
      driver-class-name: org.postgresql.Driver
  management:
    server:
      add-application-context-header: false
  endpoints:
```

web:
exposure:
include: health,info,env,metrics,beans,configprops

docker image build -t simple-api-student-main .

docker run --network app-network --name java-container -p 8081:8080
simple-api-student-main

Http server

Basics

Choose an appropriate base image.

Dockerfile

```
FROM httpd:2.4  
COPY index.html /usr/local/apache2/htdocs/
```

index.html

```
<!DOCTYPE html>  
<html>  
  <body>  
  
    <h1>My First Heading</h1>  
    <p>My first paragraph.</p>  
  
  </body>  
</html>
```

docker build -t index .

docker run -dit --network app-network --name my-running-app -p 8082:80 index

Configuration

docker exec my-running-app cat /usr/local/apache2/conf/httpd.conf > index-httpd.conf

Reverse proxy

uncomment module lines and change url proxy.

index-httpd.conf

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

```
<VirtualHost *:80>
    ProxyPreserveHost On
    ProxyPass / http://java-container:8080/
    ProxyPassReverse / http://java-container:8080/
</VirtualHost>
```

Link application

Docker-compose

[1-3 Document docker-compose most important commands](#) [1-4 Document your docker-compose file.](#)

Publish

[1-5 Document your publication commands and published images in dockerhub.](#)

```
docker tag simple-api margotfierimonte/simple-api:1.0
docker push margotfierimonte/simple-api:1.0
```

```
docker tag adminer margotfierimonte/adminer:1.0
docker push margotfierimonte/adminer:1.0
```

```
docker tag f29f2ffbae59 margotfierimonte/f29f2ffbae59:1.0
docker push margotfierimonte/f29f2ffbae59
```

Devops tp2

IMPORTANT : The docker images used for the tp2 are the one of correction.

2-1 What are testcontainers?

Testcontainers are Java libraries that allow us to run a bunch of docker containers while testing

2-2 Document your Github Actions configurations.

Création of .git/workflow folder.

add main.yml

choose the branch : main

add the jdk

add secret variable :

- DOCKERHUB_USERNAME
- DOCKERHUB_TOKEN

add the command to build and test the app :

- `cd ./devops-resources-main/solution/01-docker/simple-api/ && mvn -B verify`

add the command to log into docker :

- `docker login -u ${ secrets.DOCKERHUB_USERNAME }} -p ${ secrets.DOCKERHUB_TOKEN }}`

and finally add the part for build and push 3 images

2-3 Document your quality gate configuration.

add secret variable :

- SONAR_TOKEN

create a token with sonarCloud and get the key project

add in the command to build and test the app a part for the sonarCloud token with the keys project :

- `sonar:sonar -Dsonar.projectKey=githubaction_devops -Dsonar.organization=githubaction -Dsonar.host.url=https://sonarcloud.io -Dsonar.login=${ secrets.SONAR_TOKEN }} --file ./pom.xml`