Homework

EXERCISE 1.1: GETTING STARTED

Since we already did "Hello, World!" in the material let's do something else.

Start 3 containers from image that does not automatically exit, such as nginx, detached.

Stop 2 of the containers leaving 1 up.

Submit the output for `docker ps -a` which shows 2 stopped containers and one running

```
□ □ □ ~/work_priv/Docker_Helsinki □ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                      PORTS     NAMES
fc54de218e7b   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute           80/tcp    wonderful_antonelli
0737e48910f4   nginx     "/docker-entrypoint.…"   About a minute ago   Exited (0) 21 seconds ago             goofy_jang
7993d976cdae   nginx     "/docker-entrypoint.…"   About a minute ago   Exited (0) 21 seconds ago             cool_wu

 □ □ □ ~/work_priv/Docker_Helsinki □
```

EXERCISE 1.2: CLEANUP

We've left containers and a image that won't be used anymore and are taking space, as `docker ps -as` and `docker images` will reveal.

Clean the Docker daemon from all images and containers.

Submit the output for `docker ps -a` and `docker images`

```
□ □ □ ~/work_priv/Docker_Helsinki □ docker  ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

 □ □ □ ~/work_priv/Docker_Helsinki □



□ □ □ ~/work_priv/Docker_Helsinki □ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

 □ □ □ ~/work_priv/Docker_Helsinki □
```

EXERCISE 1.3: SECRET MESSAGE

Now that we've warmed up it's time to get inside a container while it's running!

Image `devopsdockeruh/simple-web-service:ubuntu` will start a container that outputs logs into a file. Go inside the container and use `tail -f ./text.log` to follow the logs. Every 10 seconds the clock will send you a "secret message".

Submit the secret message and command(s) given as your answer.

```
□ □ □ ~/work_priv/Docker_Helsinki □ docker run -d --rm devopsdockeruh/simple-web-service:ubuntu
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platf
7eae53318a8e524bc31c29f74fe07afb617bb24668b55dccf064baf0fc72351d

 □ □ □ ~/work_priv/Docker_Helsinki □ docker exec -it awesome_yonath bash
root@7eae53318a8e:/usr/src/app# tail -f ./text.log
Secret message is: 'You can find the source code here: https://github.com/docker-hy'
```

EXERCISE 1.4: MISSING DEPENDENCIES

Start a Ubuntu image with the process `sh -c 'while true; do echo "Input website:"; read website; echo "Searching.."; sleep 1; curl http://$website; done'`

If you're on Windows, you'll want to switch the `'` and `"` around: `sh -c "while true; do echo 'Input website:'; read website; echo 'Searching..'; sleep 1; curl http://$website; done"`.

You will notice that a few things required for proper execution are missing. Be sure to remind yourself which flags to use so that the container actually waits for input.

> Note also that curl is NOT installed in the container yet. You will have to install it from inside of the container.

Test inputting `helsinki.fi` into the application. It should respond with something like

```
<html>
  <head>
    <title>301 Moved Permanently</title>
  </head>

  <body>
    <h1>Moved Permanently</h1>
    <p>The document has moved <a href="http://www.helsinki.fi/">here</a>.</p>
  </body>
</html>
```

This time return the command you used to start process and the command(s) you used to fix the ensuing problems.

Hint for installing the missing dependencies you could start a new process with `docker exec`.

- This exercise has multiple solutions, if the curl for helsinki.fi works then it's done. Can you figure out other (smart) solutions?

```
start in terminal 1:
docker run --rm -it --name looper-it ubuntu sh -c 'while true; do echo "Input website:"; read website; echo "Searching.."; sleep 1; cu

Fix dependencies in terminal 2:
□ □ □ ~/work_priv/Docker_Helsinki □ docker exec -it looper-it bash
root@60cc074e68c9:/# apt-get update && apt-get install curl

terminal 1 check:
Input website:
helsinki.fi
Searching..
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.20.1</center>
</body>
</html>
Input website:
```

EXERCISE 1.5: SIZES OF IMAGES

In the Exercise 1.3 we used `devopsdockeruh/simple-web-service:ubuntu`.

Here is the same application but instead of Ubuntu is using Alpine Linux: `devopsdockeruh/simple-web-service:alpine`.

Pull both images and compare the image sizes. Go inside the alpine container and make sure the secret message functionality is the same. Alpine version doesn't have bash but it has sh.

```
□ □ □ ~ □ docker images -a
REPOSITORY                          TAG       IMAGE ID       CREATED       SIZE
nginx                               latest    f71a4866129b   3 days ago    135MB
ubuntu                              latest    bab8ce5c00ca   3 weeks ago   69.2MB
devopsdockeruh/simple-web-service   ubuntu    4e3362e907d5   2 years ago   83MB
devopsdockeruh/simple-web-service   alpine    fd312adc88e0   2 years ago   15.7MB

83MB vs. 15.7MB!

Secret message is: 'You can find the source code here: https://github.com/docker-hy' - still the same!
```

EXERCISE 1.6: HELLO DOCKER HUB

Run `docker run -it devopsdockeruh/pull_exercise`.

It will wait for your input. Navigate through Docker hub to find the docs and Dockerfile that was used to create the image.

Read the Dockerfile and/or docs to learn what input will get the application to answer a "secret message".

Submit the secret message and command(s) given to get it as your answer.

```
□ □ □ ~ □ docker run -it devopsdockeruh/pull_exercise
Unable to find image 'devopsdockeruh/pull_exercise:latest' locally
latest: Pulling from devopsdockeruh/pull_exercise
8e402f1a9c57: Pull complete
5e2195587d10: Pull complete
6f595b2fc66d: Pull complete
165f32bf4e94: Pull complete
67c4f504c224: Pull complete
Digest: sha256:7c0635934049afb9ca0481fb6a58b16100f990a0d62c8665b9cfb5c9ada8a99f
Status: Downloaded newer image for devopsdockeruh/pull_exercise:latest
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platf
Give me the password: basics
You found the correct password. Secret message is:
"This is the secret message"
```

EXERCISE 1.7

Dockerfile:

```
FROM ubuntu:20.04
WORKDIR /usr/src/app
COPY curl_script.sh .
RUN chmod +x curl_script.sh
RUN apt-get update && apt-get install -y curl
CMD ./curl_script.sh
```

EXERCISE 1.8

Dockerfile:

```
FROM devopsdockeruh/simple-web-service:alpine
CMD server
```

Commands:

```
□ □ □ ~/work_priv/Docker_Helsinki □ docker build . -t web_server


□ □ □ ~/work_priv/Docker_Helsinki □ docker run web_server
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platf
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env: export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /*path                    --> server.Start.func1 (3 handlers)
[GIN-debug] Listening and serving HTTP on :8080
```

EXERCISE 1.9

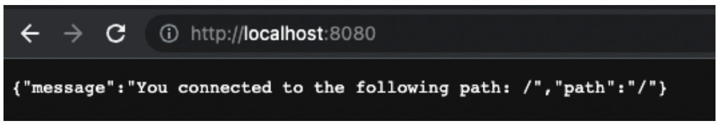
```
docker run -v "$(pwd)/from_container.log:/usr/src/app/text.log" devopsdockeruh/simple-web-service
```

EXERCISE 1.10

```
Dockerfile:
FROM devopsdockeruh/simple-web-service:alpine
CMD server

Commands:
docker run -p 8080:8080 web_server
```

Output



```
{"message":"You connected to the following path: /","path":"/"}
```

EXERCISE 1.11

```
FROM openjdk:8
EXPOSE 8080
WORKDIR /usr/src/app
COPY . .
RUN ./mvnw package
CMD ["java", "-jar", "./target/docker-example-1.1.3.jar"]
```

MANDATORY EXERCISE 1.12: HELLO, FRONTEND!

```
FROM node:16-slim
# Port change due to a newest MacOS
EXPOSE 5001
WORKDIR /usr/src/app
COPY package* .
RUN npm install
COPY . .
RUN npm run build
RUN npm install -g serve
CMD ["serve", "-s", "-l", "5001", "build"]
```

MANDATORY EXERCISE 1.13: HELLO, BACKEND!

```
FROM golang:1.16-alpine
WORKDIR /usr/src/app
COPY . .
RUN go build
ENV PORT='8080'
ENV REQUEST_ORIGIN='https://example.com'
CMD ./server


command:
docker build . -t example-be && docker run -p 8080:8080 example-be
```

MANDATORY EXERCISE 1.14: ENVIRONMENT

FE:

```
FROM node:16-slim
# Port change due to a newest MacOS
EXPOSE 5001
WORKDIR /usr/src/app
COPY package* .
RUN npm install
COPY . .
ENV REACT_APP_BACKEND_URL=http://localhost:8080/
RUN npm run build
RUN npm install -g serve
CMD ["serve", "-s", "-l", "5001", "build"]

commands:
docker build . -t example-fe && docker run -p 5001:5001 example-fe
```

BE:

```
FROM golang:1.16-alpine
EXPOSE 8080
WORKDIR /usr/src/app
COPY . .
ENV REQUEST_ORIGIN='http://localhost:5001'
RUN go build
ENV PORT='8080'
CMD ./server

commands:
docker build . -t example-be && docker run -p 8080:8080 example-be
```

EXERCISE 1.15

```
https://hub.docker.com/repository/docker/mateuszswieton/devopswithdocker/general
```

EXERCISE 1.16

I used render.com free account

I forked the repo to my own GH

I changed the App ports in the Dockerfile

I deployed is as a web-service using Render and Dockerfile

```
https://dockerops-material-fe.onrender.com/
```