## **Docker Basics**

Note: assignment 4 and 5 are of hard level and requires more time.

**Assignment-1**

Pull the ubuntu (ubuntu:latest) image and run docker run --name ubuntu-test -i -t ubuntu bash. You are now running bash inside the ubuntu image, explore the image (if you are feeling adventurous you can delete important files!). Explanation of the command: docker run starts the image, --name ubuntu-test names the container ubuntu-test and -i -t allocates an interactive pseudo TTY. To exit, type exit (you may want to delete the container AND the image).

**Assignment-2**

## Running a webserver with docker

The image prakhar1989/static-site provides a static website hostet by nginx. You need to ([Hints](https://github.com/hackundsoehne/docker-assignments##Hints-2.2)):

* pull the image
* run the image (NOT interactive, in the background and with ports exposed)
* verify it is running by opening localhost:port in the browser (if not specified docker ports X shows you the ports the container is mapped to. Also if you are running docker in a vm check the ip of the VM via docker-machine ip and need to visit ip:port)
* stop the container
* delete the container

**Assignment-3**

# Dockerfiles

Your first task is to recreate the hello-world image from above. In the folder hello\_world is a script that just prints hello word. Your task is to create an image that just runs the script hello.sh.

## 

## **Assignment-4**

## Dockerize a Java-Server

Open the Java-Project in /days\_to\_exam. Your first task is to implement the method calculateDaysToExam(String exam). Just choose one of your exams and calculate how many days are left. You can test your implementation by running main (click on the green arrow on the left) and visiting http://localhost:4567/hello for basic functionality and http://localhost:4567/exams/<exam> for the implementation of the method. After implementing the method your task is to dockerize the server. You need an environment where java is installed, so it is recommended to use �openjdk:8 image as a parent. To build the java-project run gradle fatJar, the jar is then located in /build/libs/ (use the one that contains all, if there are multiple jars). You need to Copy the jar into the image, expose port 4567 and set the entry-command to java -jar <location>. Then build the docker image using docker build <locationOfTheDockerFile>. To test the image, start the docker container detached and with open ports. See 2.2 for running webservers.

## **Assignment-5**

## Docker compose

You can get docker-compose here:

## **4.1 An frontend for the Java-Server**

Since the raw output of the Java-Server is not very beautiful (and for demonstration-purposes), we are going to add a website, that interacts with the server. You can find the project in days\_to\_exam\_frontend. You first task is to write an dockerfile to serve the html. Luckily, there are already a docker images with a servers to server static content (=just files) preinstalled. Using the image nginx is recommended, it servers all the content in /usr/share/nginx/html, so you just have to copy index.htmlinto this directory. But since we need an dynamic link to the server (depending on our enviroment it can be hosted under localhost or [www.awesomewebsite.com](http://www.awesomewebsite.com/)), we must substitute the link with an enviroment-variable. envsubst does just that, and since this command ist more complicated it is provided here: ENTRYPOINT ["/bin/bash", "-c", "envsubst < /usr/share/nginx/html/index.html > /usr/share/nginx/html/index.html && nginx -g 'daemon off;'"]. It replaces the placeholders for the enviroment variables (syntax: ${NAME\_OF\_ENV\_VAR}) and starts nginx. Your next task is to create a docker-compose file that uses has both the server and the frontend and exposes the server on port 4567 and the frontent on port 80. The frontend needs an enviroment-variable with the address of the server.

# 