Docker introguide

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.
Docker Intro: https://www.youtube.com/watch?v=_dfLOzuIg2o

To create a Docker-Image a dockerfile is needed with - among optional others - the following instructions:

`FROM <image>` → Import an exisiting image from the dockerhub to build your application on

`RUN <command>` →Run commands in order to install dependencies inside the image for your application

`COPY <source> <dest>`→Copy desired files and folders from local machine into desired location inside the image

`WORKDIR </path/to/workdir>`→ Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD instructions that follow it.

`CMD <command>` → The main purpose of a `CMD` is to provide defaults for an executing container. These defaults can include an executable, or they can omit the executable, in which case you must specify an `ENTRYPOINT` instruction as well.

https://kapeli.com/cheat_sheets/Dockerfile.docset/Contents/Resources/Documents/index

https://dockerlabs.collabnix.com/docker/cheatsheet/

1. Register on Dockerhub
To provide and manage images on dockerhub, it is necessary to register to dockerhub and prepare a repository using a name of your choice (in our case comola)

1. Go to https://hub.docker.com/ and register an account
2. After registration go to the main page and click “create repository”
3. Set the reponame and add a short description

2. To build the image from the dockerfile:
To be able to run the containerized Comola-App on different processor-architectures a multiplatform image can be generated through buildx.

1. Start docker
2. Navigate to the folder with dockerfile inside Terminal

3. `docker buildx build --load --platform linux/amd64,linux/arm64 -t <username>/<reponame>:<tag> .`
    a. `buildx`: The experimental builder. It provides the opportunity to build multiplatform images
    b. `build`: The command to build an image based on a dockerfile
    c. `--load`: After building, load the image to the registry, to make it visible for the Docker GUI and `docker images`
    d. `--platform`: Specify the platforms to build for. Since we are building on a unix platform we use linux. The arguments linux/amd64,linux/arm64 generally include all personal Laptops and desktop PCs
    e. `-t`: Specify the name of the image by using. Ideally use your login-credential on dockerhub for <username>
    f. `.`: search for a filename called “dockerfile” inside current directory. Alternatively specify the file by inserting

3. To push image into Dockerhub

When the image has been created successfully, we can push it to dockerhub to provide it for anyone.

1. Start docker
2. Login to Dockerhub using credentials for registering
    a. `Docker login -p <password> -u <username>`
3. Build image as described above
4. Docker push <username>/<reponame>:<tag>

4. To pull image from Dockerhub

When the image has been pushed successfully everyone having docker installed on it’s system can download and mount the image using the following steps

1. Start docker
2. `Docker pull <username>/<reponame>:<tag>`