**Some initial Docker knowledge**

Installation

* Docker installation. We need Docker Engine and Docker Compose, **not** Docker Desktop as such.
  + Linux or Linux VM in Windows: [https://docs.docker.com/**engine**/install/**ubuntu**/](https://docs.docker.com/engine/install/ubuntu/)

and also: <https://docs.docker.com/compose/install/other/#on-linux>

* + Windows: [https://docs.docker.com/**desktop**/install/**windows**-install/](https://docs.docker.com/desktop/install/windows-install/)
  + Mac: [https://docs.docker.com/**desktop**/install/**mac**-install/](https://docs.docker.com/desktop/install/mac-install/) (Note: Intel chip or Apple silicon?)
* The main thing is the Docker Engine (Which in Win/Mac is installed with Docker Desktop)
  + Jan 2024: v. March 2023: v. 20.10.23, Oct 2022: v. 20.10.17,
* Total process: 1. Build images, 2. Share images via registry like DockerHub (hub.docker.com), 3. Run **images** as **containers**.
* Lighter than virtual machine. Containers share the same OS resources, but are sandboxed = **isolated** from each other and the host computer **by Docker Engine** for e.g.:
  + folders and files, thus e.g. module versions - (though if you need you can share **volume**)
  + internal network ports - (though if you need you can open/**export a port** outside the container)
  + networks - (by default each container is alone, but can be put to same '**network**' if needed)

What would the VM hypervisor be then? A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing. …. Source: <https://www.vmware.com/topics/glossary/content/hypervisor.html>

* Docker images and container make it possible to run / have e.g. (BENEFITS OF DOCKERIZATION)
  + two different versions of Node on the same computer,
  + with different even 'globally' (=still actually only in that container) installed npm modules,
  + both thinking they are running 'at port 3000' (local port inside the container)
  + but those ports offered outside of the two containers as e.g. 8333 and 8444
  + containers can see each other if needed with so called **network**(s)
* With Docker we can tackle e.g these challenges (More Docker benefits):
  + Some files easily missing from the deployment => Image contains all needed
  + Software versions don't match => Image can contain all needed individual software packages, exactly the versions that are wanted
  + Environment etc. settings vary => Image always starts from its own wanted settings
  + Building the dev or runtime environment takes time => After image built, it can be taken into use very fast, like in seconds or less.
  + Building the environment is sometimes impossible to roll back to previous state => Docker allows all to be scripted, and we can improve those scripts iteratively, and even take a copy of some old script as basis for a different kind of new project.

YAML/YML format: Indentation to show nesting. Dash to show list items.  
  
<https://docs.docker.com/get-started/overview/> Docker Overview

<https://docs.docker.com/get-started/docker_cheatsheet.pdf> Basic commands to administer containers and images

<https://www.docker.com/blog/understanding-docker-networking-drivers-use-cases/>

**DOCKER VOCABULARY**

DOCKER

PaaS environment to build, deliver and run software as isolated and independent containers.

* By default containers are sand-boxed / isolated.
* You can make containers to see others by exposing ports
* You can make containers to share data also by shared volume (basically a shared folder that two or more containers can access)

DOCKER ENGINE

The engine (Dockerd daemon/process and Docker Engine API ) to manage and run the containers.

DOCKER CLIENT

The *docker* command line command. Client for giving commands about starting or stopping containers, or building, publishing images.

DOCKERFILE

Your ‘script’ for making your own Docker images. Image could be built based on source code and other assets on your disk and, if needed, ready-made images from Docker Hub or other .

DOCKERIGNORE

Says which files will not be packed into the Docker image.

DOCKER IMAGE

Ready-made from Docker Hub, or one you have created. Template that can be used to create container.

E.g. some MariaDB image you want to take into use. It’s a snapshot of a running/runnable MariaDB or other DB server that starts from a certain documented state. Typically there is a root user with known password (public information, everyone knows the password!), a certain database/schema created, like ‘test’. When taking that image into use, you must then immediately:

* secure the root user by changing the password
* create a user with less privileges with safe password or other safe access.
* give that user access to wanted schema etc.
* continue possibly with table creation, etc…

DOCKER IMAGE REGISTRY – e.g. Docker Hub

We can push/publish our images for other to use. Or pull/download images to use ourselves.

DOCKER COMPOSE

Tool for creating and starting multiple containers that talk to each other. Thus you have to make some ports exposed, or define (virtual) networks shared by multiple containers, or share volumes. You can define those in a docker-compose.yml file

DOCKER VOLUME

Persisted folder and files on disk. Allows sharing between containers, but also keeping data between container deletion and re-creation.