

# **Docker & Kubernetes**

## **Teil 2**

# Teil 2 – Installation und Nutzung

- Installation
  - Docker Desktop
  - Docker Engine
- Bedienung/Management/Kommandos

# Docker Desktop - Installation

- Windows
  - WSL 2 installieren ... „wsl --install“ in PowerShell
  - Docker Desktop laden und installieren
  - <https://docs.docker.com/desktop/install/windows-install/>
  - Anmeldung bei Docker Hub
  - Konfiguration von Docker Desktop



# Docker Desktop - Installation

- Linux
  - *Docker Desktop* laden und installieren
  - <https://docs.docker.com/desktop/install/linux-install/>
  - Anmeldung bei *Docker Hub*
  - Konfiguration von Docker Desktop

# Docker Desktop – Docker Hub

- <https://hub.docker.com>
- optional

The image shows two screenshots of the Docker Hub website. The left screenshot displays the sign-up form with fields for Username, Email, and Password, along with checkboxes for receiving updates and agreeing to terms, and a CAPTCHA verification. The right screenshot shows a search results page for Docker images, listing 'ubuntu', 'redis', and 'alpine' as top results, each with its official status, download count, and a brief description.

Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.

Get Started Today for Free

Already have an account? [Sign In](#)

Username

Email

Password

Send me occasional product updates and announcements.

I agree to the [Subscription Service Agreement](#), [Privacy Policy](#) and [Data Processing Terms](#).

Ich bin kein Roboter

[Sign Up](#)

Filters

Products

- Images
- Extensions
- Plugins

Trusted Content

- Docker Official Image
- Verified Publisher
- Sponsored OSS

Operating Systems

- Linux
- Windows

Architectures

- ARM
- ARM 64

1 - 25 of 9,515,551 available results.

 ubuntu  Updated 22 days ago

Ubuntu is a Debian-based Linux operating system based on free software.

Linux PowerPC 64 LE riscv64 IBM Z 386 x86-64 ARM ARM 64

 redis  Updated an hour ago

Redis is an open source key-value store that functions as a data structure server.

Linux Windows ARM 64 386 mips64le PowerPC 64 LE IBM Z x86-64 ARM

 alpine  Updated 14 days ago

A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in...

Linux ARM ARM 64 386 PowerPC 64 LE IBM Z riscv64 x86-64 v86-64



# Docker Desktop - Konfiguration

- Allgemeine Einstellungen für die Komponenten von Docker Desktop

The screenshot shows the Docker Desktop settings interface. The title bar says "Docker Desktop" and "Upgrade plan". The user "matthiasboldt" is logged in. A red arrow points to the "Use System Settings" radio button under the "Choose Theme for Docker Desktop" section.

**General**

- Start Docker Desktop when you log in
- Choose Theme for Docker Desktop
  - Light
  - Dark
  - Use System Settings
- Send usage statistics

Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).
- Show weekly tips
- Open Docker Dashboard at startup
- Enable Docker Compose V1/ compatibility mode

Checking this setting links '/usr/local/bin/docker-compose' to 'docker compose'. Docker Desktop ships with Docker Compose V2 only. To use Docker Compose V1,

At the bottom are "Cancel" and "Apply & Restart" buttons. The footer shows Docker status (running), RAM (12.03GB), CPU (8.75%), and a connection to the Hub. The version is v4.11.1.



# Docker Desktop - Konfiguration

- abhängig von den Ressourcen der genutzten Umgebung
- nicht zu gering wählen
- parallel genutzte Software (z.B. Dev-Umgebung) beachten

The screenshot shows the Docker Desktop settings interface. The left sidebar has sections for General, Resources (which is selected), Advanced, File sharing, Proxies, Network, Docker Engine, Experimental features, and Kubernetes. The main area is titled 'Resources' and shows three sliders for CPU, Memory, and Swap. The CPU slider is set to 6. The Memory slider is set to 12.00 GB. The Swap slider is set to 2 GB. At the bottom are 'Cancel' and 'Apply & Restart' buttons. The status bar at the bottom shows RAM 12.03GB, CPU 6.87%, Connected to Hub, v4.11.1, and a battery icon.



# Docker Desktop - Konfiguration

- Ist die Nutzung von Kubernetes nicht vorgesehen ... das auch nicht auswählen
- Startet viele Container mit Kubernetes-Basis-Diensten

The screenshot shows the Docker Desktop settings interface. The left sidebar lists several categories: General, Resources, Docker Engine, Experimental features, **Kubernetes**, Software updates, and Extensions. The Kubernetes section is currently selected, indicated by a blue highlight. On the right, under the heading "Kubernetes v1.24.2", there are two configuration options: "Enable Kubernetes" (checked) and "Show system containers (advanced)". Below these is a red-bordered button labeled "Reset Kubernetes Cluster". A tooltip at the bottom of this button states: "All stacks and Kubernetes resources will be deleted." At the bottom of the window are "Cancel" and "Apply & Restart" buttons. The status bar at the bottom displays: RAM 12.03GB, CPU 8.25%, Connected to Hub, v4.11.1, and a battery icon.



# Docker Desktop - Konfiguration

- Nützlich aber optional

The screenshot shows the Docker Desktop application window titled "Docker Desktop". The "Settings" tab is selected. On the left, there's a sidebar with icons for General, Resources, Docker Engine, Experimental features, Kubernetes, Software updates, and Extensions. The "Extensions" item is highlighted with a blue selection bar. On the right, under the "Extensions" heading, there are three configuration options:

- Enable Docker Extensions**  
Turning this option off will uninstall all extensions and disable all extension features.
- Allow only extensions distributed through the Docker Marketplace**  
This will prevent the ability to install any other extension via the Extension SDK tools.
- Show Docker Extensions system containers**  
Show Docker Desktop Extensions internal containers when using Docker commands.

At the bottom right of the window are "Cancel" and "Apply & Restart" buttons. The status bar at the bottom shows RAM usage (12.03GB), CPU usage (7.25%), and a connection to the Hub. The version number v4.11.1 is also visible.



# Docker Desktop - Konfiguration

- Dashboard zeigt, ob Docker und Kubernetes laufen
- Auswahl von Extensions

The screenshot shows the Docker Desktop application window. At the top, there are tabs for "Docker Desktop" and "Upgrade plan". On the right side of the header, there are icons for settings, user profile (matthiasboldt), and window controls. Below the header, there are links for "Containers", "Images", and "Volumes". A "Dev Environments" section is labeled "BETA". The main area is titled "tplace" with a "Give Feedback" link. It says "third-party tools within Docker Desktop to extend its functionality." with a "Learn more" link. There are buttons for "Request an extension" and "Build an extension". A dropdown menu "Sort by" is set to "Recently added". Below this, there's a section for "Extensions" also labeled "BETA". An "Add Extensions" button with a plus sign and a hand cursor icon is visible. One extension listed is "Apache Mesos" which "enables you run a local Apache Mesos cluster." with an "Install" button. Another extension listed is "PGAdmin4" which is a "management tool for PostgreSQL" with an "Install" button. At the bottom of the window, there are status indicators for Docker and Kubernetes, a "Connected to Hub" message, the version "v4.11.1", and a gear icon.



# Docker Desktop - Konfiguration

- Portainer.io
- Sinnvolle Verwaltung von Images, Containern, Volumes, ... aus dem Browser
- Alle Funktionen ... zur Ergänzung von Docker Desktop

The screenshot shows the Docker Desktop application window with the title bar "Docker Desktop" and "Upgrade plan". The main area is titled "Extensions Marketplace" with a "Give Feedback" link. It explains that Docker Extensions let you use third-party tools within Docker Desktop to extend its functionality, with a "Learn more" link. There are three listed extensions:

- Portainer** by Portainer.io: Description: "Docker container management made simple, with the world's most popular GUI-based container management platform." An "Install" button is shown.
- Slim.AI** by Slim.AI: Description: "Slim gives you the power to create secure containers faster. Deep dive into the construction of your images. Know what's in your containers." An "Install" button is shown.
- Snyk** by Snyk Ltd.: Description: "Snyk helps you identify and fix security vulnerabilities in your code before it's too late." An "Install" button is shown.

At the bottom, there are status indicators: RAM 12.03GB, CPU 2.87%, Connected to Hub, v4.11.1, and a gear icon.

# Docker Desktop - Nutzung

- enthaltene Funktionen zur Image Verwaltung
  - Pull
  - Push (zu Docker Hub)
  - Remove
  - Inspect

# Docker Desktop - Nutzung

- enthaltene Funktionen zur Container Verwaltung
    - Pause
    - Resume
    - Stop
    - Start
    - Delete
- 
- ```
graph LR; Created((Created)) -- "create" --> Running((Running)); Running -- "start" --> Running; Running -- "run" --> Running; Running -- "pause" --> Paused((Paused)); Paused -- "unpause" --> Running; Running -- "stop" --> Stopped((Stopped)); Stopped -- "start" --> Running; Stopped -- "rm" --> Deleted((Deleted)); Deleted -- "rm" --> Deleted;
```
- The diagram illustrates the state transitions for Docker containers. It shows five states: Created (light green), Running (light blue), Paused (light orange), Stopped (pink), and Deleted (grey). Transitions are as follows:
- Created → Running: start
  - Running → Running: run
  - Running → Paused: pause
  - Paused → Running: unpause
  - Running → Stopped: stop
  - Stopped → Running: start
  - Deleted → Deleted: rm (deletion)

# Docker Desktop - Nutzung

- enthaltene Funktionen zur Volume Verwaltung
  - Delete
  - Create

# Docker Engine - Installation

- Linux-Server (ubuntu 22.04) ... Vorbereitete VM für VirtualBox „Ubuntu-Docker.ova“
- WSL auf Windows 10/11 mit ubuntu-Basis
- Sammlung der Kommandos in „CMD-Docker-auf-Linux-Server.txt“

```
Ubuntu - Docker (Docker installiert) [wird ausgeführt] - Oracle VM VirtualBox : 1
Datei Maschine Anzeige Eingabe Geräte Hilfe
Client:
Context: default
Debug Mode: false
Plugins:
app: Docker App (Docker Inc., v0.9.1-beta3)
buildx: Docker Buildx (Docker Inc., v0.8.2-docker)
compose: Docker Compose (Docker Inc., v2.6.0)
scan: Docker Scan (Docker Inc., v0.17.0)

Server:
Containers: 4
Running: 0
Paused: 0
Stopped: 4
Images: 1
Server Version: 20.10.17
Storage Driver: overlay2
Backing Filesystem: extfs
Supports d_type: true
Native Overlay Diff: true
userxattr: false
Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 2
Plugins:
Volume: local
Network: bridge host ipvlan macvlan null overlay
Log: auslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 0197261a30bf81f1ee8e6a4dd2dea0ef95d67ccb
runc version: v1.1.3-0-g6724737
init version: de40ad0
Security Options:
D--Mehr--
```

```
Ubuntu - Docker (Docker installiert) [wird ausgeführt] - Oracle VM VirtualBox : 1
Datei Maschine Anzeige Eingabe Geräte Hilfe
osboxes@osboxes:~$ docker image list --all
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-world latest feb5d9fea6a5 11 months ago 18.3kB
osboxes@osboxes:~$ docker container list --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
01514ba0d002 hello-world "/hello" 6 days ago Exited (0) 6 days ago compassionate
_lishizaka
c550bdc9c053 hello-world "/hello" 6 days ago Exited (0) 6 days ago quirky_mcclintock
cb35c594a83d hello-world "/hello" 6 days ago Exited (0) 6 days ago awesome_murdoch
6c31ef2b52da hello-world "/hello" 6 days ago Exited (0) 6 days ago competent_villani
osboxes@osboxes:~$
```

# Docker Engine - Installation

- Grundlegende Pakte

```
$ sudo apt-get update  
$ sudo apt-get install ca-certificates curl gnupg lsb-release
```

- GPG key

```
$ sudo mkdir -p /etc/apt/keyrings  
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

- ubuntu Paketliste (eine Zeile)

```
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"  
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

# Docker Engine - Installation

- Docker Engine & Docker Compose installieren

```
$ sudo apt-get update  
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

- Funktionstest (einen Container ausführen)

```
$ sudo docker container run hello-world
```

- Start mit Boot konfigurieren

```
$ sudo systemctl enable docker.service  
$ sudo systemctl enable containerd.service
```

# Docker Engine - Installation

- Als Nicht-Root-User starten

```
$ sudo groupadd docker  
$ sudo usermod -aG docker $USER  
$ newgrp docker  
$ mkdir /home/"$USER"/.docker  
$ sudo chown "$USER":'$USER' /home/"$USER"/.docker -R  
$ sudo chmod g+rwx "$HOME/.docker" -R
```

- Test als Nicht-Root-User

- Funktioniert auch unter Windows

```
$ docker container run hello-world
```



```
mb@mb-GL502VMZ ~ $ docker container run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
2db29710123e: Pull complete  
Digest: sha256:7d246653d0511db2a6b2e0436cf0e52ac8c066000264b3ce63331ac66dca625  
Status: Downloaded newer image for hello-world:latest  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it to  
   your terminal.  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/
```

# Docker Engine – Portainer

- Data Volume erzeugen

```
$ docker volume create portainer_data
```

- Portainer laden & starten als Container (eine Zeile)

```
$ docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:2.14.2
```

```
$ docker container list
```

```
osboxes@osboxes:~$  
osboxes@osboxes:~$  
osboxes@osboxes:~$ docker container list  
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS     NAMES  
92dc20f1de87   portainer/portainer-ce:2.14.2   "/portainer"   9 minutes ago  Up 8 minutes  0.0.0.0:  
:8000->8000/tcp, :::8000->8000/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp, 9000/tcp   portainer  
osboxes@osboxes:~$
```





# Docker Engine – Portainer

- Start im Browser: <https://localhost:9443>
- Domain/IP-Nummer passend tauschen :-)

The image displays three screenshots of the Portainer web application running in Mozilla Firefox.

- Screenshot 1: New Portainer installation**

This screen shows the initial setup wizard. It asks for the administrator user details:

  - Username: admin
  - Password: [redacted]
  - Confirm password: [redacted]

A warning message states: "The password must be at least 12 characters long." There is a "Create user" button and a checkbox for anonymous statistics collection.
- Screenshot 2: Home**

This screen shows the environments management interface. It lists one environment named "local" which is up and running, containing 5 containers, 2 images, and 1 volume. The status is Standalone 20.10.17 /var/run/docker.sock.
- Screenshot 3: Container list**

This screen shows the list of Docker containers managed by Portainer. The table includes columns for Name, State, Quick Actions, Stack, Image, Created, and IP. The containers listed are:

| Name                   | State   | Quick Actions | Stack | Image                         | Created             | IP  |
|------------------------|---------|---------------|-------|-------------------------------|---------------------|-----|
| portainer              | running | [actions]     | -     | portainer/portainer-ce:2.14.2 | 2022-08-25 13:14:40 | 172 |
| compassionate_ishizaka | stopped | [actions]     | -     | hello-world                   | 2022-08-18 13:33:27 | -   |
| quirky_mcclintock      | stopped | [actions]     | -     | hello-world                   | 2022-08-18 13:26:19 | -   |
| awesome_murdock        | stopped | [actions]     | -     | hello-world                   | 2022-08-18 13:24:58 | -   |
| competent_villani      | stopped | [actions]     | -     | hello-world                   | 2022-08-18 13:20:49 | -   |

# Docker - CLI-Kommandos

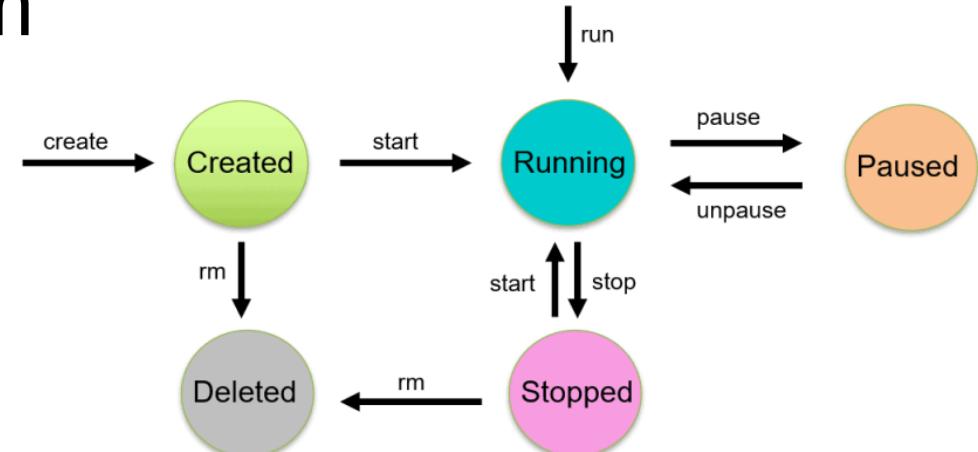
- Wesentliche Kommandos des CLI
- Übersicht im CLI selbst

```
$ docker --help
```

- Aufruf immer in der Form

```
$ docker [OPTIONS] COMMAND [SUBCOMMAND] [ARG ...]
```

- Steuerung des Zyklus



# Docker CLI

- Dokumentation  
<https://docs.docker.com/engine/reference/commandline/cli/>
- Management-Kommandos & Subkommandos (oft ohne Management-Kommandos aufrufbar => viele Dopplungen) => allgemeine Syntax:  

```
$ docker [OPTIONS] COMMAND [SUBCOMMAND] [ARG ...]
```
- Eine Auswahl der wichtigsten Kommandos ...

# Docker CLI – wichtige Kommandos

- system      Manage Docker
- image        Manage images
- container    Manage containers
- network     Manage networks
- volume      Manage volumes

# Docker CLI – *system* Kommando

- Festplattenbelegung anzeigen

```
$ docker system df
```



The screenshot shows a terminal window titled "Terminal - mb@mb-GL502VMZ ~". The window displays the Docker command-line reference and the output of the "docker system df" command.

**Docker Command-Line Reference:**

- To list available commands, either run `docker` with no arguments or `docker --help`.
- Options:**
  - `--config` string: Location of configuration file.
  - `-c, --context` string: Name of the context.
  - `-D, --debug`: Enable debug mode.
  - `--help`: Print usage information.
  - `-H, --host` value: Daemon socket path.
  - `--log-level` string: Set the log level.
  - `--tls`: Use TLS; implies `--tlscacert`.
  - `--tlscacert` string: Trust certs for TLS.
  - `--tlscert` string: Path to TLS certificate.
  - `--tlskey` string: Path to TLS key.

**Output of docker system df:**

| TYPE          | TOTAL | ACTIVE | SIZE    | RECLAIMABLE | %     |
|---------------|-------|--------|---------|-------------|-------|
| Images        | 13    | 11     | 1.45GB  | 370.7MB     | (25%) |
| Containers    | 21    | 19     | 2.294kB | 0B (0%)     |       |
| Local Volumes | 6     | 1      | 93.66kB | 0B (0%)     |       |
| Build Cache   | 0     | 0      | 0B      | 0B          |       |

# Docker CLI – *system* Kommando

- Systeminformationen anzeigen

```
$ docker system info
```



```
mb@mb-GL502VMZ ~ $ docker system info
Client:
  Context: desktop-linux
  Debug Mode: false
  Plugins:
    app: Docker App (Docker Inc., v0.9.1-beta3)
    buildx: Docker Buildx (Docker Inc., v0.8.2)
    compose: Docker Compose (Docker Inc., v2.6.1)
    extension: Manages Docker extensions (Docker Inc., v0.2.8)
    sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
    scan: Docker Scan (Docker Inc., v0.17.0)

Server:
  Containers: 21
  Running: 19
  Paused: 0
  Stopped: 2
  Images: 13
  Server Version: 20.10.17
  Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  usernxattr: false
  Logging Driver: json-file
  Cgroup Driver: cgroups
  Cgroup Version: 2
  Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
    Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
  Swarm: inactive

$ docker system info [OPTIONS]
```

The terminal window has a title bar 'Terminal - mb@mb-GL502VMZ ~'. The main area displays the command output. A lightbulb icon is overlaid on the top right of the terminal window.

**Usage**

**Options**

# Docker CLI – *system* Kommando

- Speicherplatz freigeben / aufräumen



```
$ docker system prune
```

```
Terminal - mb@mb-GL502VMZ ~
```

Datei Bearbeiten Ansicht Terminal Reiter Hilfe

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ reference / Command-line reference / Docker CLI (docker) / docker system / docker syste

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker system events

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker system info

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker system prune

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker tag

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker top

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker trust

mb@mb-GL502VMZ ~ \$

mb@mb-GL502VMZ ~ \$ docker system prune

WARNING! This will remove:

- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

Are you sure you want to continue? [y/N] y

Deleted Containers: docker volume

ea0bd226781fe7aa39e149f1d8d9aa61e918de7c79b2e73644957a5ceb2e53bb  
994189da0688bf8dc18ce2ce57c8661e17d40384961e5c615cd9bd1f8fd1d6e

Total reclaimed space: 0B

mb@mb-GL502VMZ ~ \$ Compose CLI reference

# DOCKER SYSTEM INFO

Display system-wide information

## Usage

```
$ docker system info [OPTIONS]
```

## Options

# Docker CLI – *image* Kommando

- Ein Image untersuchen

```
$ docker image inspect <image name>
```

```
$ docker image inspect hello-world
```



```
mb@mb-GL502VMZ ~ $ docker image inspect hello-world
[{"Id": "sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412",
 "RepoTags": ["hello-world:latest"],
 "RepoDigests": [
     "hello-world@sha256:7d246653d0511db2a6b2e0436cf0e52ac8c066000264b3ce63331ac66dca625"
 ],
 "Parent": "",
 "Comment": "lock, tag",
 "Created": "2021-09-23T23:47:57.442225064Z",
 "Container": "8746661ca3c2f215da94e6d3f7dfdcfaf5ec0b21c9aff6af3dc379a82fb72",
 "ContainerConfig": {
     "Hostname": "8746661ca3c2",
     "Domainname": "",
     "User": "",
     "AttachStdin": false,
     "AttachStdout": false,
     "AttachStderr": false,
     "Tty": false,
     "OpenStdin": false,
     "StdinOnce": false,
     "Env": [
         "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
     ],
     "Cmd": [
         "/bin/sh",
         "-c",
         "#(nop)",
         "CMD [\"/hello\"]"
     ]
 }
```

# Docker CLI – *image* Kommando

- Images auflisten

```
$ docker image list  
$ docker image list --all
```



A screenshot of a terminal window titled "Terminal - mb@mb-GL502VMZ ~". The window shows the output of the "docker image list --all" command. The output lists various Docker images along with their details such as tag, image ID, creation date, and size. The terminal also displays the Docker documentation for "docker system info" and "Options".

| REPOSITORY                                              | TAG                                                                        | IMAGE ID       | CREATED       | SIZE   |
|---------------------------------------------------------|----------------------------------------------------------------------------|----------------|---------------|--------|
| portainer/portainer-docker-extension                    | 2.14.2                                                                     | ab836adaa325   | 4 weeks ago   | 278MB  |
| docker/disk-usage-extension                             | 0.2.5                                                                      | 5f8f804dbfa2   | 5 weeks ago   | 2.79MB |
| hubproxy.docker.internal:5000/docker/desktop-kubernetes | kubernetes-v1.24.2-cni-v1.1.1-critools-v1.24.2-cri-dockerd-v0.2.1-1-debian | 5dcc4b79ec39   | 2 months ago  | 364MB  |
| k8s.gcr.io/kube-apiserver                               | v1.24.2                                                                    | d3377ffb7177   | 2 months ago  | 130MB  |
| k8s.gcr.io/kube-controller-manager                      | v1.24.2                                                                    | 34cdf99b1bb3   | 2 months ago  | 119MB  |
| k8s.gcr.io/kube-proxy                                   | v1.24.2                                                                    | a634548d10b0   | 2 months ago  | 110MB  |
| k8s.gcr.io/kube-scheduler                               | v1.24.2                                                                    | 5d725196c1f4   | 2 months ago  | 51MB   |
| k8s.gcr.io/etcd                                         | 3.5.3-0                                                                    | aeebe758cef4c  | 4 months ago  | 299MB  |
| k8s.gcr.io/pause                                        | 3.7                                                                        | 221177c6082a   | 5 months ago  | 711KB  |
| k8s.gcr.io/coredns/coredns                              | v1.8.6                                                                     | a4ca41631cc7   | 10 months ago | 46.8MB |
| hello-world                                             | latest                                                                     | feb5d9fe6a5    | 11 months ago | 13.3kB |
| docker/desktop-vpnkit-controller                        | v2.0                                                                       | 8c2c38aa676e   | 15 months ago | 21MB   |
| docker/desktop-storage-provisioner                      | v2.0                                                                       | I 99f89471f470 | 16 months ago | 41.9MB |

# Docker CLI – *image* Kommando

- Ein Image entfernen

```
$ docker image rm --force <image name>
```

```
$ docker image rm --force hello-world
```



A screenshot of a Docker desktop interface showing a terminal window. The terminal shows a history of Docker commands, followed by the command `docker image rm --force hello-world`. The output indicates that the image was untagged and then deleted, with the final message showing the SHA-256 hash of the deleted image.

```
mb@mb-GL502VMZ ~ $ docker image push
mb@mb-GL502VMZ ~ $ docker image rm
mb@mb-GL502VMZ ~ $ docker image save
mb@mb-GL502VMZ ~ $ docker image tag
mb@mb-GL502VMZ ~ $ docker images
mb@mb-GL502VMZ ~ $ docker import
mb@mb-GL502VMZ ~ $ docker info
mb@mb-GL502VMZ ~ $ docker inspect
mb@mb-GL502VMZ ~ $ docker kill
mb@mb-GL502VMZ ~ $ docker rm
mb@mb-GL502VMZ ~ $ docker image rm --force hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:7d246653d0511db2a6b2e0436cf0e52ac8c066000264b3ce63331ac66dca625
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
Deleted: sha256:e07ee1baac5fae6a26f30cabfe54a36d3402f96afda318fe0a96cec4ca393359
mb@mb-GL502VMZ ~ $
```

# Docker CLI – *image* Kommando

- Ein Image von Registry laden

```
$ docker image pull <image name>
```

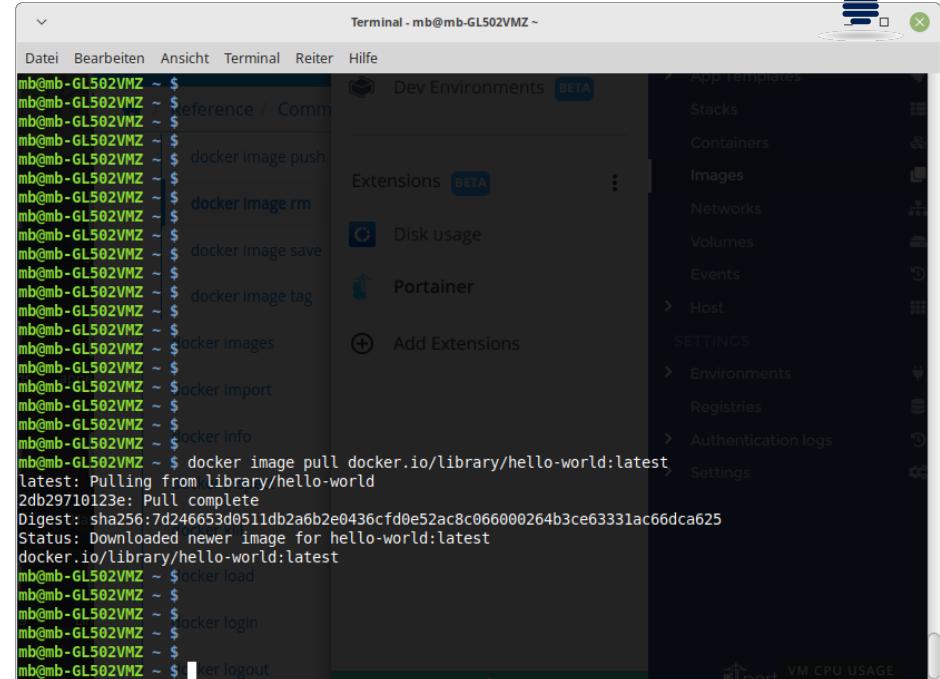
```
$ docker image pull docker.io/library/hello-world:latest
```

<image name>



- Registry-URL
- Name des Images
- ..
- ..
- Tag

bei Nutzung von Docker Hub kann die Registry-URL entfallen



```
mb@mb-GL502VMZ ~ $ docker image pull <image name>
mb@mb-GL502VMZ ~ $ docker image pull docker.io/library/hello-world:latest
mb@mb-GL502VMZ ~ $ latest: Pulling from library/hello-world
mb@mb-GL502VMZ ~ $ Digest: sha256:7d246653d0511db2a6b2e0436cf0e52ac8c066000264b3ce63331ac66dca625
mb@mb-GL502VMZ ~ $ Status: Downloaded newer image for hello-world:latest
mb@mb-GL502VMZ ~ $ docker.io/library/hello-world:latest
mb@mb-GL502VMZ ~ $ docker load
mb@mb-GL502VMZ ~ $ docker login
mb@mb-GL502VMZ ~ $ docker logout
mb@mb-GL502VMZ ~ $ docker logout
mb@mb-GL502VMZ ~ $
```

# Docker CLI – *image* Kommando

- Ein Image in Registry speichern

```
$ docker image push <image name>
```

```
$ docker image push registry.gitlab.com/xxx/test:latest
```

- Zuvor Tag anlegen

```
$ docker image SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
```

```
$ docker -l fatal image tag registry.gitlab.com/xxx/test:v123 registry.gitlab.com/xxx/test:latest"
```

# Docker CLI – *image* Kommando

- Ein Image als Datei speichern

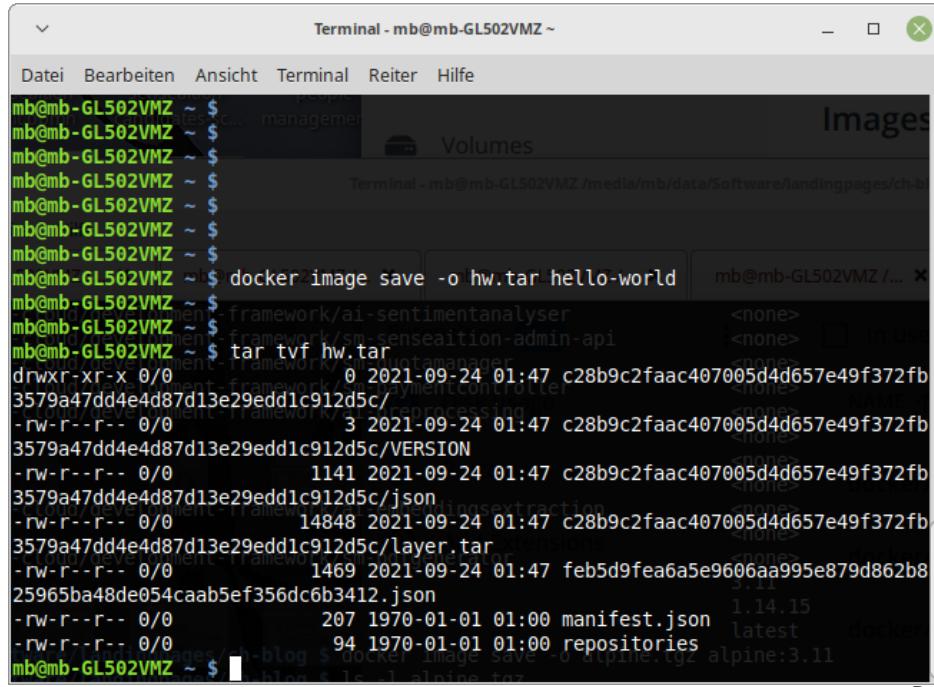
```
$ docker image save <image name>
$ docker image save -o <tar-file name> <image name>
```

```
$ docker image save -o hw.tar hello-world
```

```
$ docker image save hello-world > hw.tar
```

Docker schreibt das Abbild des Images in ein TAR-Archiv.

Ohne „-o“ wird auf die Standardausgabe STDOUT geschrieben und mit „-o“ wird die spezifizierte Datei erzeugt.



```
mb@mb-GL502VMZ ~ $ docker image save -o hw.tar hello-world
mb@mb-GL502VMZ ~ $ tar tvf hw.tar
drwxr-xr-x 0/0 0 2021-09-24 01:47 c28b9c2faac407005d4d657e49f372fb
3579a47dd4e4d87d13e29edd1c912d5c/
-rw-r--r-- 0/0 3 2021-09-24 01:47 c28b9c2faac407005d4d657e49f372fb
3579a47dd4e4d87d13e29edd1c912d5c/VERSION
-rw-r--r-- 0/0 1141 2021-09-24 01:47 c28b9c2faac407005d4d657e49f372fb
3579a47dd4e4d87d13e29edd1c912d5c/json
-rw-r--r-- 0/0 14848 2021-09-24 01:47 c28b9c2faac407005d4d657e49f372fb
3579a47dd4e4d87d13e29edd1c912d5c/layer.tar
-rw-r--r-- 0/0 1469 2021-09-24 01:47 febd9fea6a5e9606aa995e879d862b8
25965ba48de054caab5ef356dc6b3412.json
-rw-r--r-- 0/0 207 1970-01-01 01:00 manifest.json
-rw-r--r-- 0/0 94 1970-01-01 01:00 repositories
mb@mb-GL502VMZ ~ $
```



# Docker CLI – *image* Kommando

- Ein Image aus Datei laden

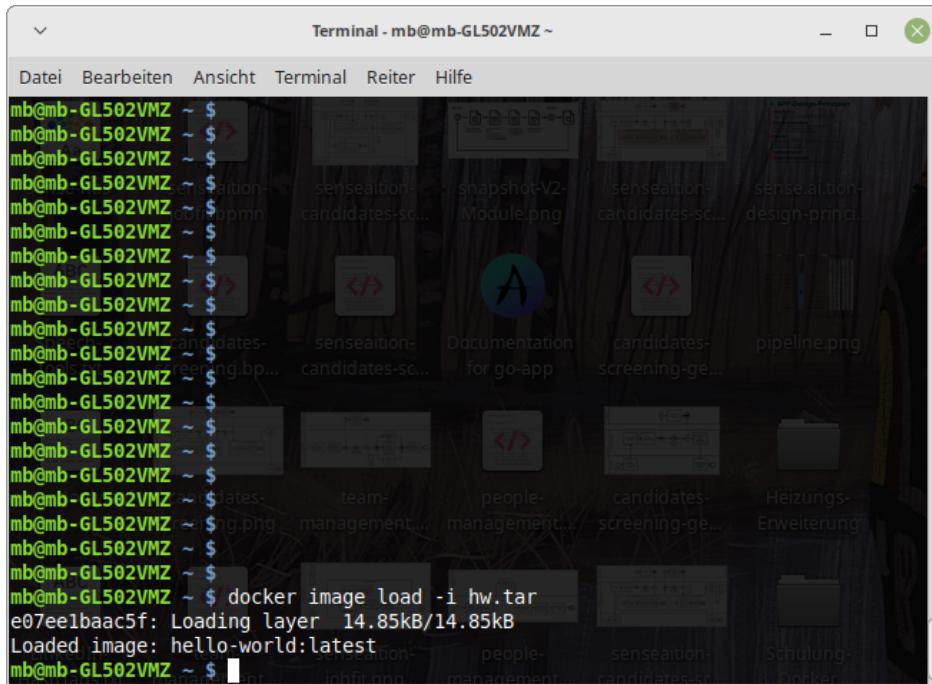
```
$ docker image load  
$ docker image load -i <tar-file name>
```

```
$ docker image rm hello-world  
$ docker image load -i hw.tar
```

```
$ docker image rm hello-world  
$ cat hw.tar | docker image load
```

Docker liest ein Abbild eines Images aus einem TAR-Archiv.

Ohne „-i“ wird von der  
Standardeingabe STDIN gelesen.



# Docker CLI – *image* Kommando

- Ein Image erzeugen/bauen

```
$ docker image build [OPTIONS] PATH | URL | -  
$ docker image build --tag <image name>:<image tag> .  
$ docker image build -file <Dockerfile-name> --tag <image name>:<image tag> .
```

```
$ docker image build --tag simple .  
$ cat Dockerfile | docker image build --tag simple -
```



Docker erstellt anhand eines „Dockerfile“ ein Image. Die Datei „Dockerfile“ wird im Verzeichnis PATH erwartet. Mit Option „--file“ kann ein anderer Dateiname angegeben werden.

```
# sehr einfaches Dockerfile  
FROM alpine:latest  
CMD ["echo", "Das ist ein sehr einfacher Container..."]
```

```
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker image build --tag simple .  
[+] Building 4.7s (6/6) FINISHED  
=> [internal] load build definition from Dockerfile  
=> transferring dockerfile: 37B  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [internal] load metadata for docker.io/library/alpine:latest  
=> [auth] library/alpine:pull token for registry-1.docker.io  
=> [1/1] FROM docker.io/library/alpine:latest@sha256:bc41182d7ef5ffc53a4  
=> => resolve docker.io/library/alpine:latest@sha256:bc41182d7ef5ffc53a4  
=> => sha256:bc41182d7ef5ffc53a40b044e725193bc10142a1243 1.64kB / 1.64kB  
=> => sha256:1304f174557314a7ed9eddab4eab12fed12cb0cd9809e4c2 528B / 528B  
=> => sha256:9c6f0724472873bb50a2ae67a9e7adc57673a183ce 1.47kB / 1.47kB  
=> => sha256:213ec9aee27d8be045c6a92b7eac22c9a64b4455819 2.81MB / 2.81MB  
=> => extracting sha256:213ec9aee27d8be045c6a92b7eac22c9a64b44558193725a  
=> => exporting to image  
=> => exporting layers  
=> => writing image sha256:98f2cb63d424bf4e0034954d9465b54ce7e9b9c2c1b41  
=> => naming to docker.io/library/simple  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
```

# Docker CLI – *Dockerfile*

- Dockerfile (Wiederholung :-)
  - Text-Datei
  - Folge aus Anweisungen zum Aufbau des Images
  - Ist extrem vielseitig und flexibel nutzbar
  - Jede Anweisung erzeugt ein Layer im Image
  - Anweisung in Form: **INSTRUCTION arguments**
  - Kommentare mit „#“ am Anfang der Zeile
  - <https://docs.docker.com/engine/reference/builder/>
  - Wesentliche Anweisungen ...

# Docker CLI – Dockerfile - Beispiel

```
# ai-audioanalyser
FROM alpine:latest

# installation of necessary certificates for ssl communication
RUN apk --no-cache add ca-certificates

# installation of service module for the API within the home directory of the image
ADD ai-audioanalyser /home

# only necessary for running container within docker, kubernetes manages its own
environment
ENV SMLOGLEVEL 100
ENV SMHOST 0.0.0.0
ENV SMPORT 4242
ADD cluster.json /home
ADD queue.json /home
ADD keywords.csv /home
ADD classifier.csv /home

# start program/process
STOPSIGNAL SIGTERM
EXPOSE ${SMPORT}
ENTRYPOINT /home/ai-audioanalyser
```

# Docker CLI – *Dockerfile*

FROM <image name>:<tag>

- Setzt das „base image“
- Alle weiteren Build-Schritte beziehen sich darauf
- Festlegung der Plattform möglich
- Alpine Linux, Ubuntu, ... Windows

FROM alpine:latest

# Docker CLI – *Dockerfile*

RUN <command>

- Führt ein Kommando in einem neuen Layer aus
- Das Ergebnis wird integriert in das Image

```
RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'  
RUN apk --no-cache add ca-certificates  
RUN cd /src && CGO_ENABLED=0 go build -o /root/client ./main.go
```

# Docker CLI – *Dockerfile*

```
WORKDIR <directory>
```

- Arbeitsverzeichnis (im Container) setzen
- Für Kommandos
  - RUN
  - CMD
  - ENTRYPOINT
  - COPY
  - ADD

```
WORKDIR /home
```

# Docker CLI – *Dockerfile*

ADD <src> <desc>

- Kopiert von <src> nach <desc>
- Kopiert werden Dateien, Verzeichnisse (auch von URLs) von <src>
- Kopiert wird nach <desc> im Image
- Endet <desc> mit „/“, wird in das Verzeichnis kopiert, sonst wird eine Datei geschrieben (falls <desc> kein Verzeichnis ist)
- Kann TAR-Archive automatisch entpacken

```
ADD build/mybinary /home/
ADD http://domain.com/config.txt /home/config/
ADD data/* /home/data/
```

# Docker CLI – *Dockerfile*

COPY <src> <desc>

- Wie ADD, aber
  - Kein Abruf von Dateien von URL's
  - Kein automatisches Entpacken von TAR-files
- *best practice* ist Nutzung von COPY

```
COPY build/mybinary /home/
COPY data/* /home/data/
```

# Docker CLI – *Dockerfile*

VOLUME <directory>

- Benennt einen Mount-Point für externe Volumes

VOLUME /external-data

# Docker CLI – *Dockerfile*

ENV <key>=<value>

- Setzen von Umgebungsvariablen
- 12 Faktoren => Parametrisierung von Programmen über Umgebungsvariablen
- Umgebungsvariablen können beim Start eines Containers überschrieben werden

```
$ docker container run --env <key>=<value> <image>
```

```
ENV LOGLEVEL=100  
ENV HOST=0.0.0.0  
ENV PORT=4242
```

# Docker CLI – *Dockerfile*

`EXPOSE <port>`

`EXPOSE <port>/<protocol>`

- Teilt Docker mit, welche Ports der Container nach außen nutzt
- Können bei Start automatisch publiziert werden
  - \$ docker container run <image> --publish-all
- Überschreiben beim Start

`EXPOSE 4242  
EXPOSE 80/udp`

```
$ docker container run <image> --publish <host-port>:<container-port>  
$ docker container run <image> --publish 80:4242  
$ docker container run <image> --publish 443:443
```

# Docker CLI – Dockerfile

```
CMD <command> <para-1> ... <para-n>
```

```
CMD [<command>], [<para-1>], ..., [<para-n>]
```

```
CMD [<para-1>], ..., [<para-n>]
```

- Standardprogramm des Containers
- Kommando wird bei Start des Containers ausgeführt
- Erste Form wird in shell gestartet (von „/bin/sh -c“ aus)
- Dritte Form dient der Parameterübergabe an ENTRYPOINT
- Dockerfile darf nur ein einziges CMD enthalten

```
CMD ["echo", "Das ist ein sehr einfacher Container..."]  
CMD echo "Das ist ein sehr einfacher Container..."
```

# Docker CLI – Dockerfile

```
ENTRYPOINT <command> <para-1> ... <para-n>
```

```
ENTRYPOINT [<>“<command>”], [<>“<para-1>”], ..., [<>“<para-n>”]
```

- Standardprogramm, das bei Start des Containers ausgeführt wird
- Mit ENTRYPOINT kann ein Container als Executable ausgeführt werden
- Erste Form wird in shell gestartet (von „/bin/sh -c“ aus)
- Dockerfile darf nur ein einziges ENTRYPOINT enthalten

```
ENTRYPOINT ["echo", "Das ist ein sehr einfacher Container..."]  
ENTRYPOINT echo "Das ist ein sehr einfacher Container..."
```

# Docker CLI – Dockerfile

- Entwurf eines eigenen Dockerfiles
  - „Dockerfile-1.test“
  - Umgebungsvariable MYVAR auf „1“ setzen
  - Container beim Start/Lauf gibt kurzen Text plus Wert von MYVAR aus
- ... editieren ... ausprobieren

```
$ docker image build --tag testimage --file Dockerfile-1.test .
$ docker image build -t testimage -f Dockerfile-1.test .
```

```
$ docker container run testimage
$ docker container run --env MYVAR=42 testimage
```

- ... nach 15 Minuten Vergleich



```
Terminal - mb@mb-GL502VMZ ~/Desktop/Schulung-Docker
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container run --env MYVAR=42 testimage
Mein Container-Test '42'
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
```

# Docker CLI – *container* Kommando

- Einen container erzeugen

```
$ docker container create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

```
$ docker container create --env MYVAR=42 --name testcontainer testimage
```



# Wichtigste Optionen

- --name
  - --env, -e
  - --interactive, -i
  - --network
  - --publish
  - --volumn, -v
  - --tmpfs
  - --tty, -t

```
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container create --env MYVAR=42 --name testcontainer testimage  
3705e9ea83e23538858c9a8ad0536be35b6cb58a679e879986744bf486eae40a  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
```

# Docker CLI – *container* Kommando

- Option von „create“

--name <container-name>

- Erzeugt einen Container unter dem angegebenen Namen
- In weiter Kommandos kann der Namen genutzt werden

--env <key>=<value>

-e <key>=<value>

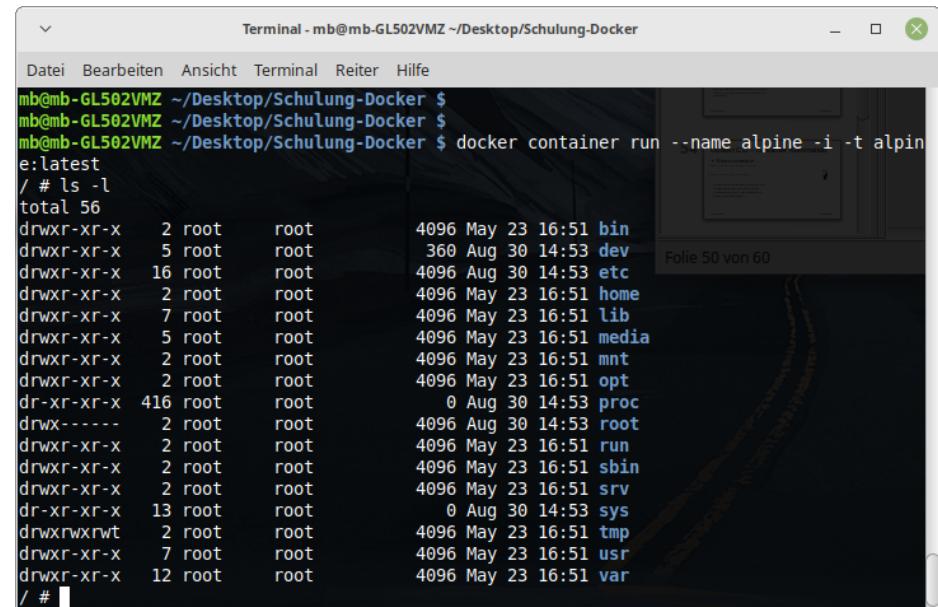
- Setzt/überschreibt die Umgebungsvariable beim Start des Containers
- Kann auch Liste sein

--interactive, -i

- Verbindung zu STDIN des Containers wird von Console erzeugt

--tty, -t

- Ein pseudo-tty nutzen (Console)



The screenshot shows a terminal window titled "Terminal - mb@mb-GL502VMZ ~/Desktop/Schulung-Docker". The terminal output is as follows:

```
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ 
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container run --name alpine -i -t alpine:latest
/ # ls -l
total 56
drwxr-xr-x  2 root    root      4096 May 23 16:51 bin
drwxr-xr-x  5 root    root      360 Aug 30 14:53 dev
drwxr-xr-x 16 root    root     4096 Aug 30 14:53 etc
drwxr-xr-x  2 root    root      4096 May 23 16:51 home
drwxr-xr-x  7 root    root      4096 May 23 16:51 lib
drwxr-xr-x  5 root    root      4096 May 23 16:51 media
drwxr-xr-x  2 root    root      4096 May 23 16:51 mnt
drwxr-xr-x  2 root    root      4096 May 23 16:51 opt
dr-xr-xr-x 416 root    root      0 Aug 30 14:53 proc
drwx----- 2 root    root      4096 Aug 30 14:53 root
drwxr-xr-x  2 root    root      4096 May 23 16:51 run
drwxr-xr-x  2 root    root      4096 May 23 16:51 sbin
drwxr-xr-x  2 root    root      4096 May 23 16:51 srv
dr-xr-xr-x 13 root    root      0 Aug 30 14:53 sys
drwxrwxrwt  2 root    root      4096 May 23 16:51 tmp
drwxr-xr-x  7 root    root      4096 May 23 16:51 usr
drwxr-xr-x 12 root    root      4096 May 23 16:51 var
/ #
```

# Docker CLI – *container* Kommando

- Option von „create“

- network=<network name>

- Verbindung des Containers mit dem angegebenen Netzwerk

- publish [<host-ip>:]<host-port>:<container-port>[/<protocol>]

- p [<host-ip>:]<host-port>:<container-port>[/<protocol>]

- Publizieren/Binden eines Ports
    - \$ docker container run -p 127.0.0.1:80:8080/tcp testcontainer
    - \$ docker container run -p 80:8080 testcontainer

- volume [<volume name>:]<host-directory>:<container-directory>

- v [<volume name>:]<host-directory>:<container-directory>

- Bindet ein Verzeichnis des Hosts in den Container ein

- tmpfs <container-directory>

- Nutzung eines temporären Verzeichnisses im RAM des Hosts
  - Nur bei Linux-Hosts
  - Werden nicht zwischen Containern geteilt

# Docker CLI – *container* Kommando

- Einen container starten

```
$ docker container start [OPTIONS] <container name>
```

```
$ docker container start testcontainer
```



Startet einen erzeugten und/oder gestoppten Container.

# Docker CLI – *container* Kommando

- Einen container erzeugen & starten

```
$ docker container run [OPTIONS] IMAGE [COMMAND] [ARG...]
```



```
$ docker container run --env MYVAR=42 testimage
```

„run“ führt nacheinander „create“ und „start“ aus

Die Parameter entsprechen denen von „create“.

# Docker CLI – *container* Kommando

- Einen container anhalten

```
$ docker container stop [OPTIONS] CONTAINER  
$ docker container kill [OPTIONS] CONTAINER
```

Stoppt einen laufenden Container.

„stop“ sendet ein SIGTERM und „kill“ ein SIGKILL an den laufenden Prozess im Container.

# Docker CLI – *container* Kommando

- Einen container löschen

```
$ docker container rm [OPTIONS] CONTAINER
```

```
$ docker container run --name alpine --rm -i -t -d alpine top  
$ docker container rm --force testcontainer
```



Mit --force können auch laufende Container gelöscht werden.

# Docker CLI – *container* Kommando

- Ein zus. Programm in einem container starten

```
$ docker container exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

```
$ docker container run --name alpine --rm -i -t -d alpine top  
$ docker container exec alpine ls -l
```



Das Programm/Kommando wird nur in einem laufenden Container ausgeführt.

The screenshot shows a terminal window titled "Terminal - mb@mb-GL502VMZ ~/Desktop/Schulung-Docker". The terminal output is as follows:

```
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container run --name alpine --rm -i -t -d alpine top  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container exec alpine ls -l  
total 56  
drwxr-xr-x 2 root root 4096 May 23 16:51 bin  
drwxr-xr-x 5 root root 360 Aug 30 19:33 dev  
drwxr-xr-x 16 root root 4096 Aug 30 19:33 etc  
drwxr-xr-x 2 root root 4096 May 23 16:51 home  
drwxr-xr-x 7 root root 4096 May 23 16:51 lib  
drwxr-xr-x 5 root root 4096 May 23 16:51 media  
drwxr-xr-x 2 root root 4096 May 23 16:51 mnt  
drwxr-xr-x 2 root root 4096 May 23 16:51 opt  
dr-xr-xr-x 416 root root 0 Aug 30 19:33 proc  
drwx----- 2 root root 4096 May 23 16:51 root  
drwxr-xr-x 2 root root 4096 May 23 16:51 run  
drwxr-xr-x 2 root root 4096 May 23 16:51 sbin  
drwxr-xr-x 2 root root 4096 May 23 16:51 srv  
dr-xr-xr-x 13 root root 0 Aug 30 19:33 sys  
drwxrwxrwt 2 root root 4096 May 23 16:51 tmp  
drwxr-xr-xd. 7 root root 4096 May 23 16:51 usr  
drwxr-xr-x 12 root root 4096 May 23 16:51 var  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
```

# Docker CLI – container Kommando

- Logs eines Containers zeigen

```
$ docker container logs [OPTIONS] CONTAINER
```

```
$ docker container run --name alpine --rm -i -t -d alpine sh -c "while true; do ls; sleep 3; done"  
$ docker container logs -f alpine
```



Die Option „-f“ folgt dem Ende der Logs  
... STDOUT.

```
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container run --name alpine --rm -i -t -d alpine sh -c "while true; do ls; sleep 3; done"  
e8f7ebdb0c3b65d1219514222df24dc8154cf0e0779af98839796b8935f3b6ae  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container logs -f alpine  
bin etc lib mnt proc run sbin srv tmp var  
dev home media opt root sbin sys usr  
bin etc lib mnt proc run srv tmp var  
dev home media opt root sbin sys usr  
bin etc lib mnt proc run sbin srv tmp var  
dev home media opt root sbin sys usr  
bin etc lib mnt proc run sbin sys usr  
dev home media opt root sbin sys usr  
bin etc lib mnt proc run sbin sys usr  
dev home media opt root sbin sys usr  
bin etc lib mnt proc run sbin sys usr  
dev home media opt root sbin sys usr  
bin etc lib mnt proc run sbin sys usr  
dev home media opt root sbin sys usr  
^C  
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $
```

# Docker CLI – *container* Kommando

- Ressourcenverbrauch der Container zeigen

```
$ docker container stats [OPTIONS] [CONTAINER...]
```

```
$ docker container run --name alpine --rm -i -t -d alpine sh -c "while true; do ls; sleep 3; done"  
$ docker container stats  
$ docker container rm --force alpine
```

Zeigt Verbrauch der Ressourcen des Hosts und von eingestellten Limits.

Beenden mit String+C



| Terminal - mb@mb-GL502VMZ ~/Desktop/Schulung-Docker |            |                    |           |             |                |       |
|-----------------------------------------------------|------------|--------------------|-----------|-------------|----------------|-------|
| Datei                                               | Bearbeiten | Ansicht            | Terminal  | Reiter      | Hilfe          | CPU % |
| CONTAINER ID                                        | NAME       | MEM USAGE / LIMIT  | MEM %     | NET I/O     | BLOCK I/O      | PIDS  |
| 96f544d61b6e                                        | alpine     | 1.43MiB / 11.76iB  | 0.01%     | 1.18kB / 0B | 1.41MB / 0B    | 2     |
| 2ed9d27caeef                                        | portainer  | 9.703MiB / 11.76iB | 0.08%     | 1.39kB / 0B | 30.2MB / 344kB | 8     |
|                                                     |            |                    |           |             |                |       |
|                                                     |            | CREATED            | SIZE      |             |                |       |
|                                                     |            | 22 days ago        | 5.54 MB   |             |                |       |
|                                                     |            | over 1 year ago    | 41.85 MB  |             |                |       |
|                                                     |            | over 1 year ago    | 21.03 MB  |             |                |       |
|                                                     |            | about 1 month ago  | 2.79 MB   |             |                |       |
|                                                     |            | months ago         | 364.07 MB |             |                |       |
|                                                     |            | 11 months ago      | 46.83 MB  |             |                |       |

```
CPU % 0.00% 1.418MiB / 11.76iB 0.01%  
0.00% 9.703MiB / 11.76iB 0.08%  
REMOTE REPOSITORIES
```

# Docker CLI – *container* Kommando

- Prozesse eines Containers zeigen

```
$ docker container top CONTAINER
```

```
$ docker container run --name alpine --rm -i -t -d alpine sh -c "while true; do ls; sleep 3; done"
$ docker container top alpine
$ docker container rm --force alpine
```



Zeigt die in einem Container laufenden Prozesse an.

```
mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ docker container run --name alpine --rm -i -t -d alpine sh -c "while true; do ls; sleep 3; done" 90bf7b15e0029e125c0acbb0cccd01dc4f27ffae445fcf521d1af4a91ba37315de mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ docker container top alpine UID PID PPID C STIME over ITTYar ago TIME MB CMD mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ root 30001 29973 0 06:35 ? 00:00:00 sh -c while true; do ls; sleep 3; done root about 1 month 30207 2.79 MB 30001 0 06:35 ? 00:00:00 sleep 3 mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ mb@mb-GL502VMZ ~Desktop/Schulung-Docker $ docker container rm --force alpine alpine mb@mb-GL502VMZ ~Desktop/Schulung-Docker $
```

# Docker CLI – *container* Kommando

- Details eines Containers zeigen

```
$ docker container inspect CONTAINER
```

```
$ docker container run --name alpine --rm -i -t -d alpine sh -c "while true; do ls; sleep 3; done"
$ docker container inspect alpine
$ docker container rm --force alpine
```



Zeigt detaillierte Informationen über einen Container an ... z.B. zum Netzwerk.

```
Terminal - mb@mb-GL502VMZ ~/Desktop/Schulung-Docker
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
mb@mb-GL502VMZ ~/Desktop/Schulung-Docker $ docker container inspect alpine
[{"Id": "cecf90c1cf79498be9b012570ad9c4d0e461c7e60e560b4bad7103a43c9eb79a", "Created": "2022-08-31T06:39:19.941503938Z", "Path": "sh", "Args": ["-c", "while true; do ls; sleep 3; done"], "Size": 5.54 MB, "State": {"Status": "running", "Running": true, "Paused": false, "Restarting": false, "OOMKilled": false, "Dead": false, "Pid": 34266, "ExitCode": 0, "Error": "", "StartedAt": "2022-08-31T06:39:20.276082278Z", "FinishedAt": "0001-01T00:00:00Z"}, "Image": "sha256:9c6f0724472873bb50a2ae67a9e7adcb57673a183cea8b06eb778dca859181b5"}]
```

# ... Bildquellen

- RedHat
- IDG
- Docker
- Earthly
- Aquasec
- dev.to