

How to docker?

# Docker something

Julia Winkler

19.06.2024

# Gliederung

Einführung

Dockerfile

Einfache Container

# Disclaimer

- Bei weitem nicht alles
- Nur Allgemeine Auszüge



commands im Terminal  
ausführen  
Ordner `examples` , wenn  
nichts da steht



CodeTour (VSCode Plugin)  
Code im Repo

# Wiso, Weshalb, Warum?

Why use Docker?

**Trusted by developers.  
Chosen by Fortune 100 companies.**

Docker provides a suite of development tools, services, trusted content, and automations, used individually or together, to accelerate the delivery of secure applications.

# Wiso, Weshalb, Warum?

Why use Docker?

**Trusted by developers.  
Chosen by Fortune 100 companies.**

Docker provides a suite of development tools, services, trusted content, and automations, used individually or together, to accelerate the delivery of secure applications.

”a sandboxed process on your machine that is isolated from all other processes on the host machine”

# Wiso, Weshalb, Warum?

Why use Docker?

**Trusted by developers.  
Chosen by Fortune 100 companies.**

Docker provides a suite of development tools, services, trusted content, and automations, used individually or together, to accelerate the delivery of secure applications.

”a sandboxed process on your machine that is isolated from all other processes on the host machine”

”It works on my computer”

# Wiso, Weshalb, Warum?

Why use Docker?

**Trusted by developers.  
Chosen by Fortune 100 companies.**

Docker provides a suite of development tools, services, trusted content, and automations, used individually or together, to accelerate the delivery of secure applications.

"a sandboxed process on your machine that is isolated from all other processes on the host machine"

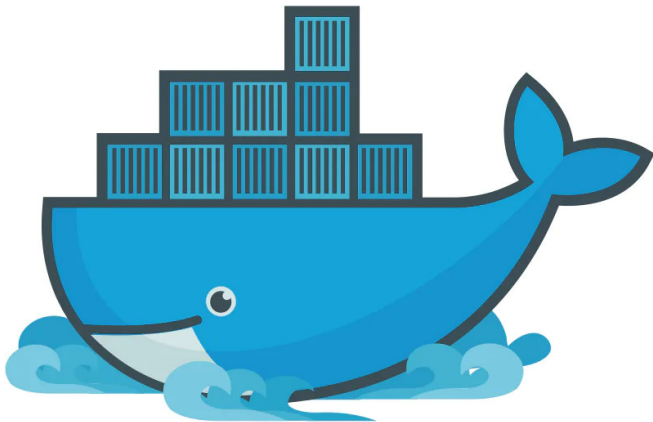
"It works on my computer"

"faster onboarding and testing while also simplifying the deployment of services"



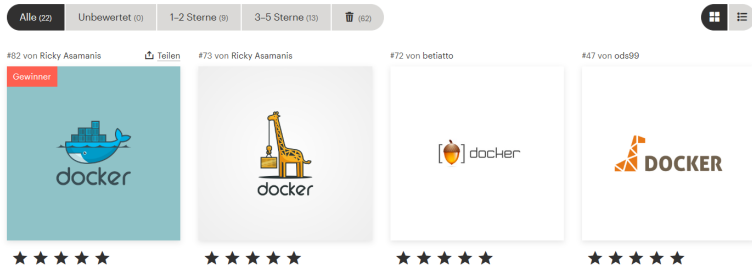
Wer ist Moby Dock?

Wer ist Moby Dock?



# Wer ist Moby Dock?

## Entwürfe



Wettbewerb zum Icon für Docker

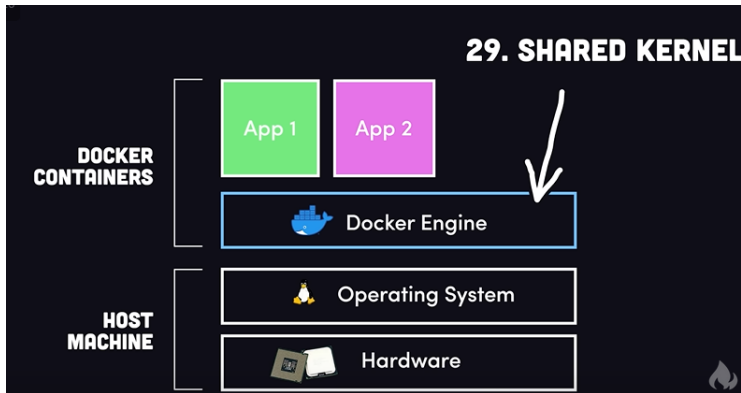
# Was ist Docker?

## Docker

freie Software zur Isolierung von Anwendungen

Containervirtualisierung

"light weight" Virtual Maschine



# Wichtige Begriffe

## Container

Umgebung in der die tatsächliche Anwendung läuft

## Image

Blaupausen, um einen Container zu erstellen

## Dockerfile

Anleitung, um ein Image zu erstellen

## Registry

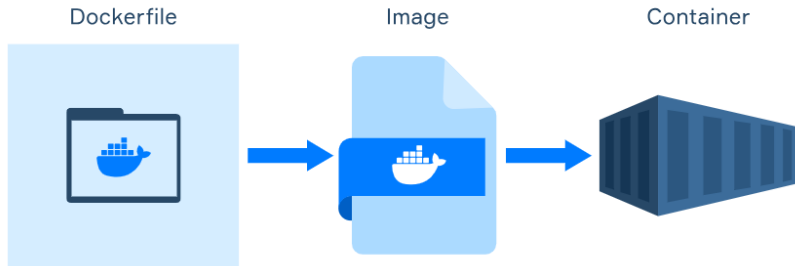
z.B. Docker Hub, EAC.... Ort an dem viele verschiedene Images gespeichert und geteilt werden können

## Docker Compose

Orchestrierungstool für Dockerfile

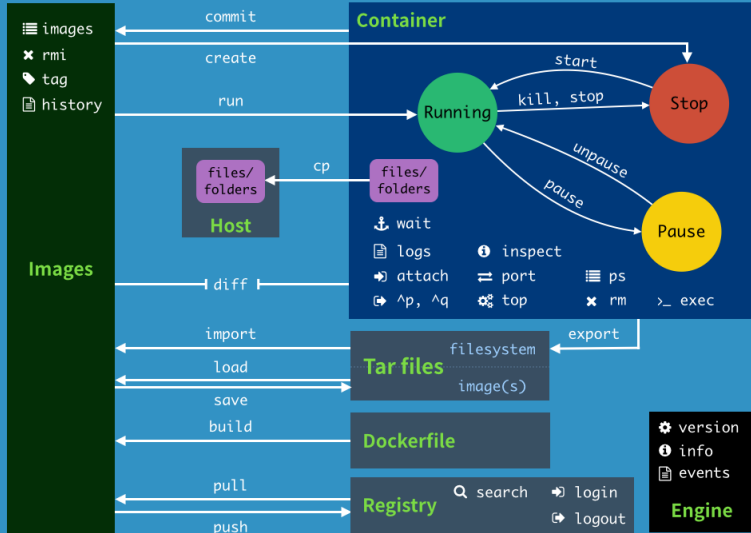
Wrapper für einen oder mehrere Container

# Zusammenhang der Docker Komponenten



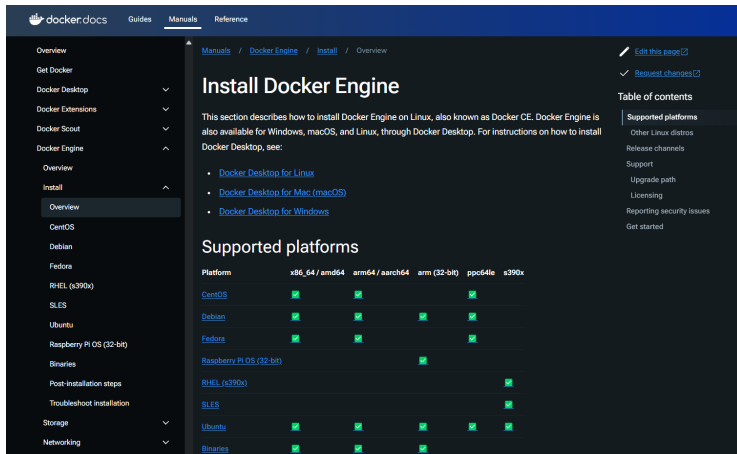
# Zusammenhang der Docker Komponenten

## Docker Commands Diagram



@fntsrlike

# Wie kriege ich dieses "Docker"?



The screenshot shows the Docker documentation website. The left sidebar contains a navigation menu with categories like 'Overview', 'Get Docker', 'Docker Desktop', 'Docker Extensions', 'Docker Scout', 'Docker Engine', 'Overview', 'Install', 'Overview', 'CentOS', 'Debian', 'Fedora', 'RHEL (s390x)', 'SLES', 'Ubuntu', 'Raspberry Pi OS (32-bit)', 'Binaries', 'Post-installation steps', 'Troubleshoot installation', 'Storage', and 'Networking'. The main content area is titled 'Install Docker Engine' and includes a paragraph explaining that Docker Engine is also available for Windows, macOS, and Linux through Docker Desktop. Below this, there are links for 'Docker Desktop for Linux', 'Docker Desktop for Mac (macOS)', and 'Docker Desktop for Windows'. A 'Supported platforms' section follows, featuring a table with columns for Platform, x86\_64 / amd64, arm64 / aarch64, arm (32-bit), ppc64le, and s390x. The table lists various operating systems and their compatibility status, indicated by green checkmarks.

Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	ppc64le	s390x
<a href="#">CentOS</a>	✓	✓		✓	
<a href="#">Debian</a>	✓	✓	✓	✓	
<a href="#">Fedora</a>	✓	✓		✓	
<a href="#">Raspberry Pi OS (32-bit)</a>			✓		
<a href="#">RHEL (s390x)</a>					✓
<a href="#">SLES</a>					✓
<a href="#">Ubuntu</a>	✓	✓	✓	✓	✓
<a href="#">Binaries</a>	✓	✓	✓		

Doku

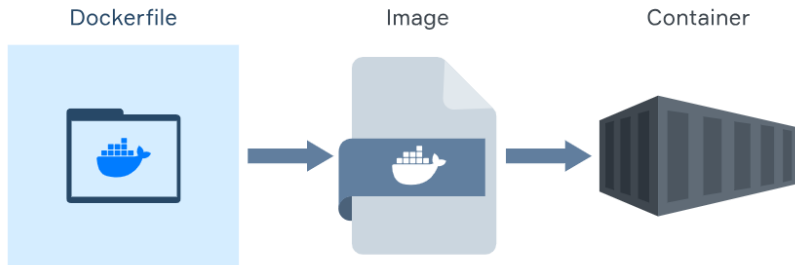


# Hello World



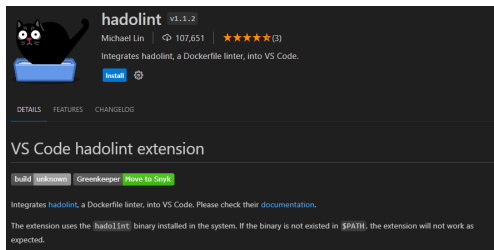
```
> docker -v  
> docker --help  
> docker run hello-world
```

# Zusammenhang der Docker Komponenten



# Dockerfile

- Ein Dockerfile ist die Anleitung um ein Image zu erstellen.
- Standardmäßig heißt die Datei 'Dockerfile'
- weitere Optionen mit `docker buildx build`



Plugin für die Arbeit mit Docker

# docker build command

```
docker build [OPTIONS] PATH | URL | -
```

Build an image from a Dockerfile

[OPTIONS]

**-f, --file string** Name of the Dockerfile (default:  
"PATH/Dockerfile")

**-t, --tag stringArray** Name and optionally a tag (format:  
"name:tag")

**PATH** in most cases .

Beispiele:

```
docker build . # 'Dockerfile' im aktuellen Ordner
```

```
docker build -t myimage:v1 .
```

```
docker build -f Docker.cmd .
```

```
docker build ./examples/FastAPI/Dockerfile
```



Beispiel Dockerfile :

---

```
FROM alpine:lastest
```

```
CMD [ "echo", "Hello World" ]
```

---

Weitere Informationen und Instruction

<https://docs.docker.com/reference/dockerfile/>

# CMD vs. ENTRYPOINT



```
> docker build -t example:cmd -f Dockerfile.cmd .  
> docker build -t example:entry -f Dockerfile.entry .
```

## CMD vs. ENTRYPOINT



```
> docker build -t example:cmd -f Dockerfile.cmd .  
> docker build -t example:entry -f Dockerfile.entry .  
> docker run example:cmd  
> docker run example:cmd hello  
> docker run example:entry hello
```

# CMD vs. ENTRYPOINT



```
> docker build -t example:cmd -f Dockerfile.cmd .  
> docker build -t example:entry -f Dockerfile.entry .  
> docker run example:cmd  
> docker run example:cmd hello  
> docker run example:entry hello
```

- beide definieren den, was nach Container start ausgeführt wird
- `CMD` kann überschrieben werden
- `ENTRYPOINT` bestimmt den command, neue Parameter werden angehängen



# RUN

```
> docker build -t example:single -f Dockerfile.single .  
> docker build -t example:multi -f Dockerfile.multi .
```

*# Entstandene Images anschauen*

```
> docker ps -a  
> docker images
```

# RUN

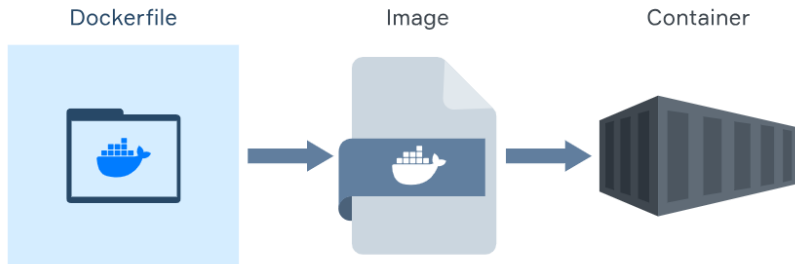
```
> docker build -t example:single -f Dockerfile.single .
> docker build -t example:multi -f Dockerfile.multi .
```

*# Entstandene Images anschauen*

```
> docker ps -a
> docker images
```

- pro `RUN` baut Docker einen Layer
- Layer werden gecached und nach Möglichkeit wiederverwendet
- versucht `RUN` instructions zu verbinden
- verbessert built-time und Image gröÙe

# Zusammenhang der Docker Komponenten



## docker run command

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

Create and run a new container from image

[OPTIONS]

- d** Detach from terminal — run in background
- e** Set environment variables
- it** Interactive terminal, enter container terminal
- mount mount** Attach a filesystem mount to the container
- p [host]:[port]** Publish a container's port(s) to the host
- P** Publish all exposed ports
- rm** Automatically remove the container when it exits
- v, -volume list** Bind mount a volume
- w** Provides an execution directory inside the container

# Python bsp

---

```
FROM alpine
```

```
# Exec form
```

```
CMD ["echo", "Hello World."]
```

```
#shell form
```

```
CMD echo Hello Students
```

---

# Volumes



# React

---

```
FROM alpine
```

```
# Exec form
```

```
CMD ["echo", "Hello World."]
```

```
#shell form
```

```
CMD echo Hello Students
```

---

# Multistage builds

- 1



# React - Multistage

---

```
FROM alpine
```

```
# Exec form
```

```
CMD ["echo", "Hello World."]
```

```
#shell form
```

```
CMD echo Hello Students
```

---

## Dockerfile Best practices

- RUN commands
- Order of COPY
- Volumes
- Multistage

# Docker Compose

- Vorteile
- UseCases

# Docker Compose zu Python

---

```
FROM alpine
```

```
# Exec form
```

```
CMD ["echo", "Hello World."]
```

```
#shell form
```

```
CMD echo Hello Students
```

---

# Docker Compose Webapp

---

```
FROM alpine
```

```
# Exec form
```

```
CMD ["echo", "Hello World."]
```

```
#shell form
```

```
CMD echo Hello Students
```

---

# title

- OpenDrone Map

es Arbeit

wi arbeit

me work

# Cheatsheet

- `docker run`
- `docker build`
- `docker push`, `pull`
- `docker ps -a`
- `docker rm / rmi`
- ...

## Cooler Quellen und so weiter

- <https://www.docker.com/>
- Offizielle Dokumentation:  
<https://docs.docker.com/get-started/>
-