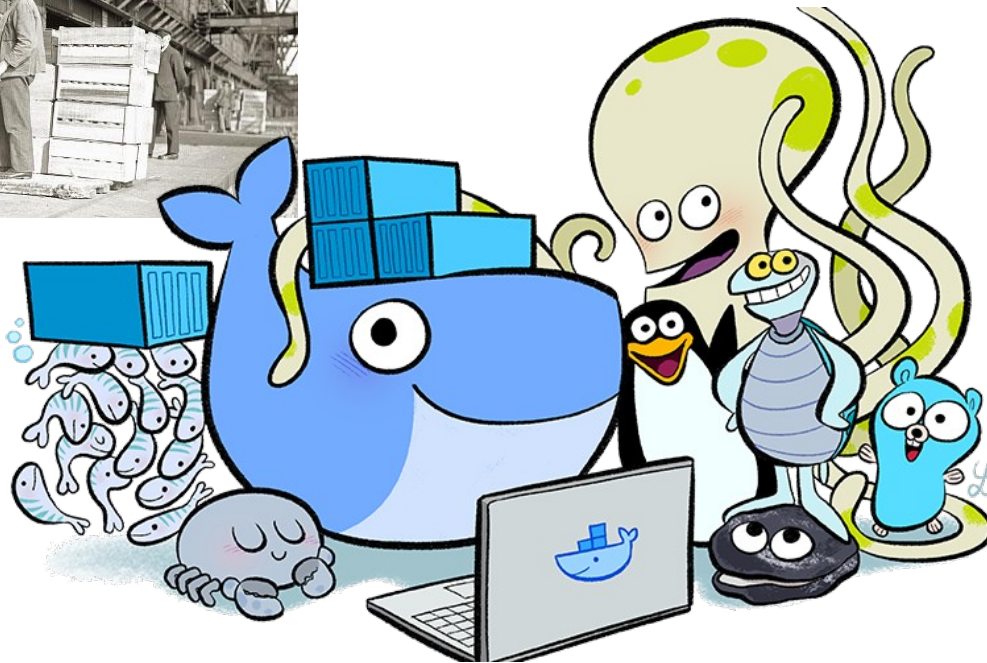
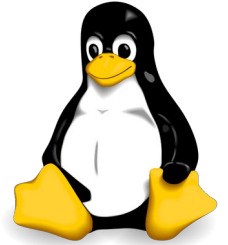


LUG Albtal

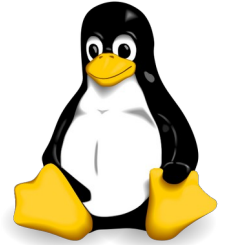




LUG Albtal

Was ist Docker?

- Erstellen, Ausführen, Verwalten von Software-Container
- Open-Source
- läuft unter Linux
- Firma (Docker Inc.)

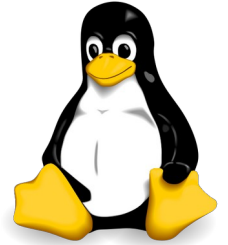


LUG Albtal

Technisch:

Gruppe von Prozessen, die isoliert voneinander ausgeführt werden

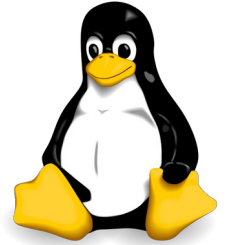
- Software-Container
- **Containervirtualisierung**



LUG Albtal

Basis sind bekannte Techniken:

- chroot
- cgroups
- Namespaces
- UnionFS (Union file system)

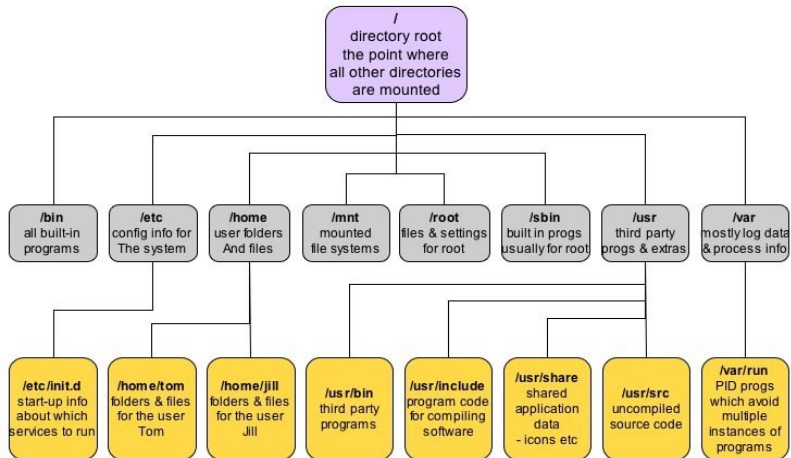


LUG Albtal

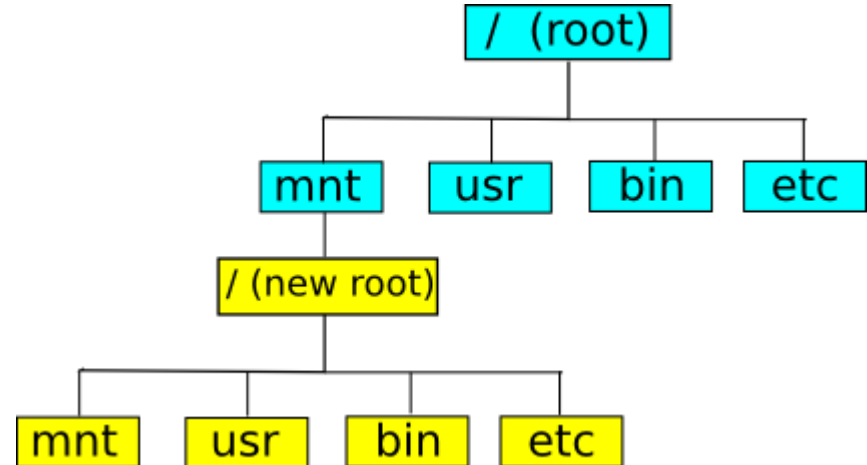
chroot

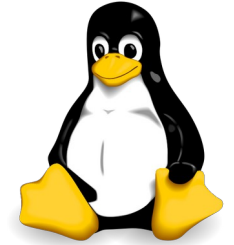
- „change root“: root-Verzeichnis im Unterordner

A Typical Linux File System



chroot

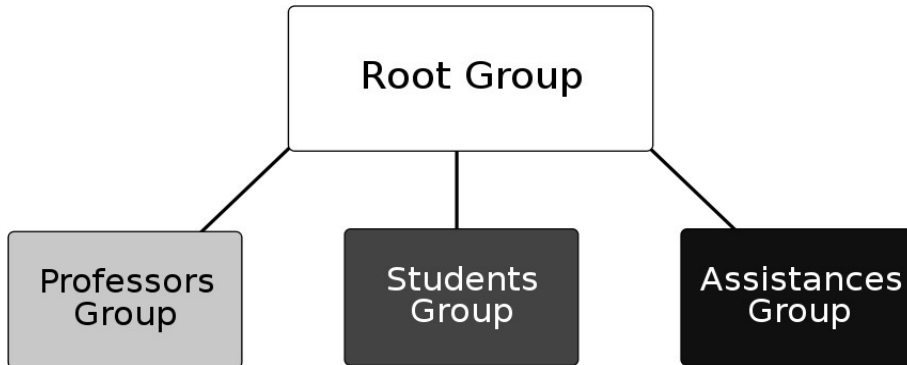




LUG Albtal

cgroups

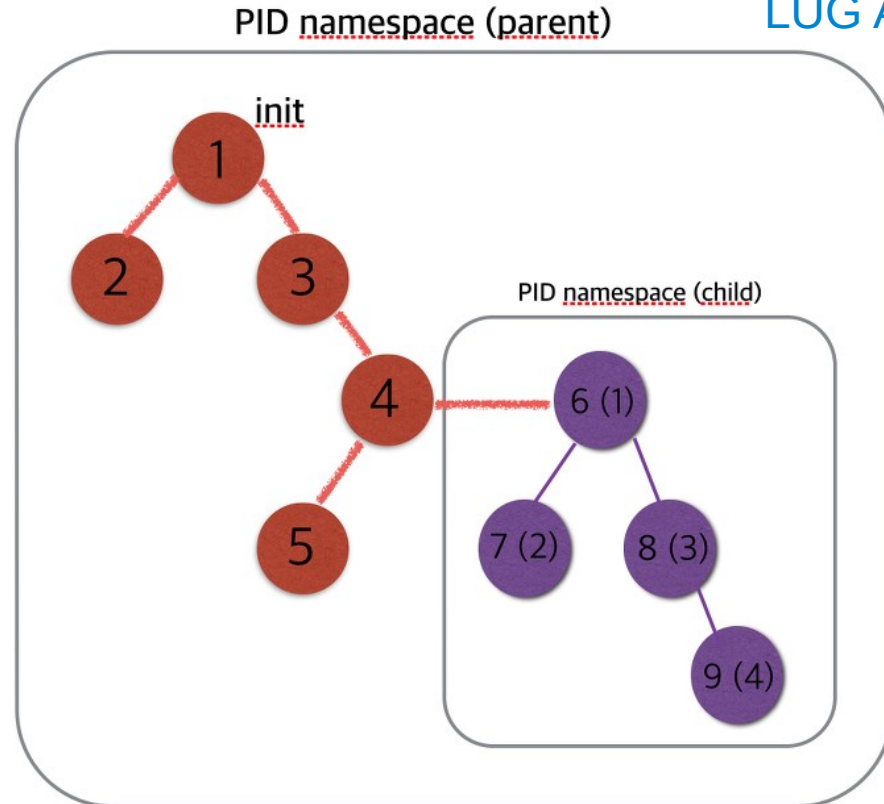
- Prozessgruppen
- OS verwaltet Ressourcen

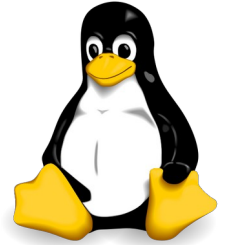


Name	PID	Nutzer %	System %	Priorität	Vm-Größe	VmRss	Benutzer	Befehl
init	1	0,00	0,00	0	2 036	632	root	init [2]
dcopserver	17138	0,00	0,00	0	26 080	2 944	bifo	dcopserver [kdeinit] --nosid
kaccess	17162	0,00	0,00	0	27 248	9 688	bifo	kaccess [kdeinit]
kded	17142	2,00	0,50	0	43 044	18 176	bifo	kded [kdeinit] --new-startup
kdeinit	17135	0,00	0,00	0	26 040	5 360	bifo	kdeinit Running...
kio_file	17160	0,00	0,00	0	26 416	7 100	bifo	kio_file [kdeinit] file /tmp/ksocket-bifo/kd
kio_file	17797	0,00	0,00	0	29 044	7 336	bifo	kio_file [kdeinit] file /tmp/ksocket-bifo/kd
kio_file	17798	0,00	0,00	0	29 044	7 336	bifo	kio_file [kdeinit] file /tmp/ksocket-bifo/kd
kio_file	17799	0,00	0,00	0	29 044	7 332	bifo	kio_file [kdeinit] file /tmp/ksocket-bifo/kd
kio_file	17800	0,00	0,00	0	29 044	7 328	bifo	kio_file [kdeinit] file /tmp/ksocket-bifo/kd
klauncher	17140	0,00	0,00	0	29 420	8 784	bifo	klauncher [kdeinit] --new-startup
konqueror	17790	0,00	0,00	0	41 704	23 604	bifo	konqueror [kdeinit] --silent
kwin	17152	0,00	0,00	0	29 376	12 732	bifo	kwin [kdeinit] -session 1012dc6d3d3000
kdesktop	17154	0,00	0,00	0	33 580	16 664	bifo	kdesktop [kdeinit]
kdm	3647	0,00	0,00	0	3 052	676	root	/usr/bin/kdm
kdm	16974	0,00	0,00	0	4 044	1 440	root	-.1
startkde	17034	0,00	0,00	0	5 136	1 568	bifo	/bin/sh
kicker	17156	0,00	0,00	0	34 256	15 780	bifo	kicker [kdeinit]
klipper	17177	0,00	0,00	0	28 316	11 692	bifo	klipper [kdeinit]
kmix	17172	0,00	0,00	0	30 644	14 720	bifo	kmix [kdeinit] -session 1012dc6d3d3000
ksmserver	17151	0,00	0,00	0	27 236	9 992	bifo	ksmserver [kdeinit]
start_kdeinit	17131	0,00	0,00	0	1 512	160	bifo	start_kdeinit
syndock	17174	0,00	0,00	0	36 608	12 368	bifo	syndock [kdeinit]

Namespaces

- Isoliert Prozessgruppen voneinander

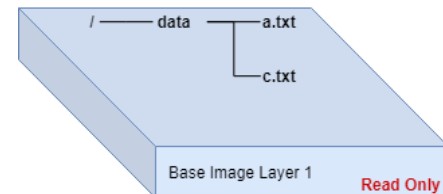
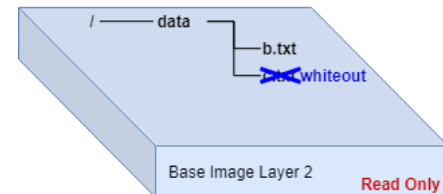
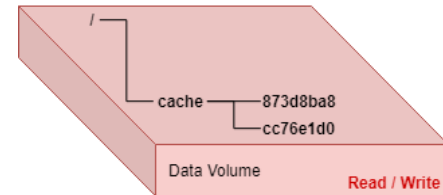
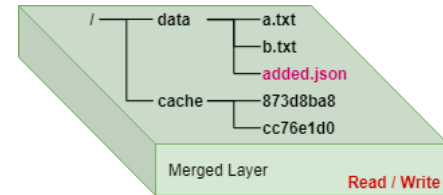


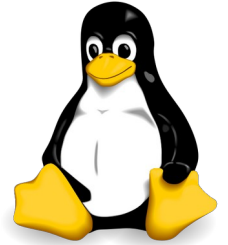


LUG Albtal

UnionFS

- Mehrere Dateisysteme werden vereinigt
- Dateien und Ordner werden überlagert

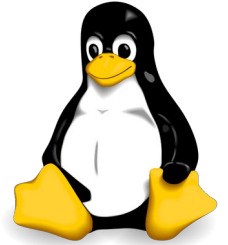




LUG Albtal

Containervirtualisierung ↔ Virtualisierung

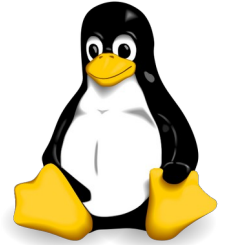
- Containervirtualisierung nutzt Kernel und Teile des Gastsystems
- Hypervisoren (KVM, VMware, VirtualBox) sind vollwertige Betriebssysteminstanzen
- Beide Virtualisierungstechniken isolieren Systeme voneinander



LUG Albtal

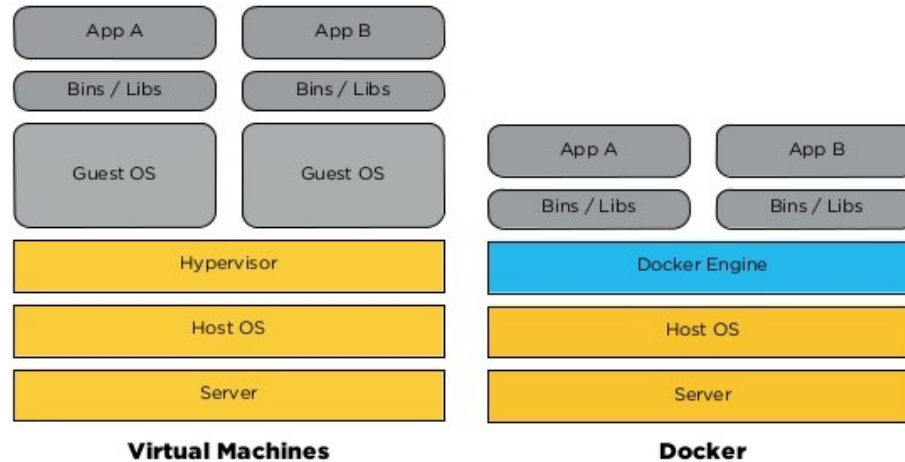
Containervirtualisierung ↔ Virtualisierung

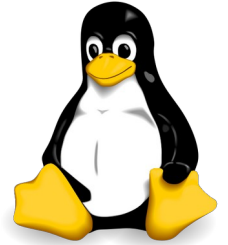
- Virtualisierung ist die Bereitstellung eines gesamten Systems (Betriebssystem + Programme)
- Container sind einzelne, isolierte Programme
→ *Microservices*



LUG Albtal

Containervirtualisierung ↔ Virtualisierung





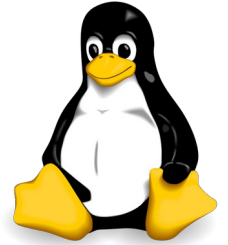
LUG Albtal

Begriffe:

- **Base Image:** Vorgefertigtes Image. Enthält alles Notwendige aber nicht mehr.
- **Image:** Vorlage für Container. Enthält Bibliotheken und Schnittstellen zum Betriebssystem. Enthält Programme
- **Dockerfile:** Datei, die ein Image erzeugt
- **Container:** Image, das ausgeführt wird

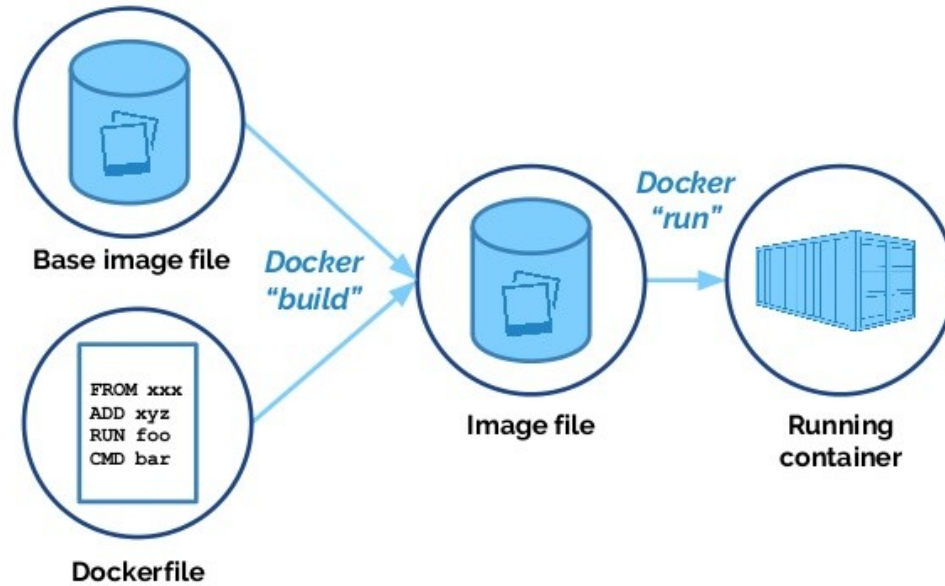


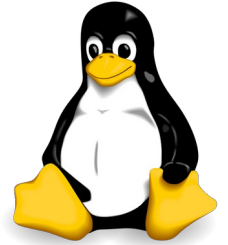
docker



LUG Albtal

Docker images and containers





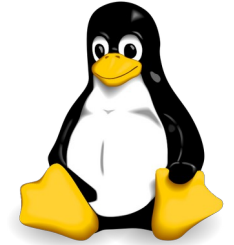
LUG Albtal

Installation von Docker unter Ubuntu:

```
sudo apt-get install docker.io
```

Installation testen:

```
sudo docker run hello-world
```



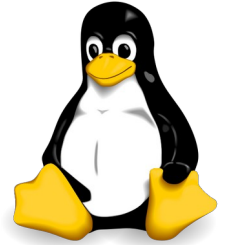
LUG Albtal

Fertige Images von Docker hub laden

The screenshot shows a web browser window displaying the Docker Hub search results page. The URL is https://hub.docker.com/search?type=image&operating_system=linux. The page shows a list of Docker images filtered by 'Linux'. The results include:

- Oracle Database Enterprise Edition** (DOCKER CERTIFIED) by Oracle, updated 2 years ago. Tags: Container, Docker Certified, Linux, x86-64, Databases.
- Oracle Database 12c Enterprise Edition** by Oracle.
- Oracle Java 8 SE (Server JRE)** (DOCKER CERTIFIED) by Oracle, updated 3 months ago. Tags: Container, Docker Certified, Linux, x86-64, Programming Languages.
- couchbase** (OFFICIAL IMAGE) updated 3 minutes ago. 10M+ Downloads, 472 Stars. Tags: Container, Linux, x86-64, Storage, Application Frameworks.
- Oracle WebLogic Server** (DOCKER CERTIFIED) by Oracle, updated 3 months ago.

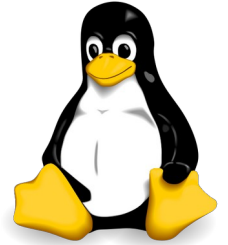
The left sidebar contains filters for 'Docker Certified', 'Verified Publisher', and 'Official Images'. Below these are category filters such as 'Analytics', 'Application Frameworks', 'Application Infrastructure', 'Application Services', 'Base Images', 'Databases', 'DevOps Tools', 'Featured Images', 'Messaging Services', 'Monitoring', 'Operating Systems', 'Programming Languages', 'Security', and 'Storage'.



LUG Albtal

Beispiel:

```
volker@volker-VirtualBox:~$ docker run debian echo 'TEST'  
Unable to find image 'debian:latest' locally  
latest: Pulling from library/debian  
4ae16bd47783: Pull complete  
Digest: sha256:2f04d3d33b6027bb74ecc81397abe780649ec89f1a2af18d7022737d0482cefe  
Status: Downloaded newer image for debian:latest  
TEST  
volker@volker-VirtualBox:~$
```

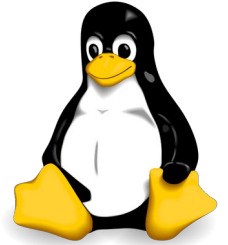
LUG Albtal

Aufgabe:

Acrobat Reader als Container

Was wird benötigt:

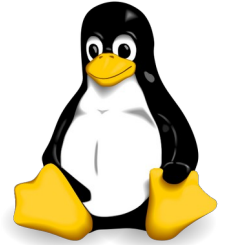
- Pakete i386 (da Acrobat Reader nur in 32-Bit verfügbar)
- Paket Acrobat Reader
- Paket CUPS (Linux-Druck-System)



LUG Albtal

Ablauf:

1. Basis-Image wählen
2. Ordner /acoread erstellen
3. Datei /acoread/Dockerfile erstellen
4. Image erstellen



LUG Albtal

Aufbau Dockerfile:

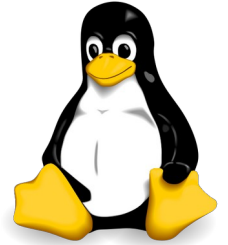
```
<Docker - Kommando> <Linux - Kommando>
```

Beispiel:

```
RUN apt-get update
```

RUN: Docker-Kommando

apt-get update: Linux-Kommando



LUG Albtal

Datei Dockerfile:

```
FROM ubuntu:18.04
```

→ Basis-Image ist Ubuntu 18.04

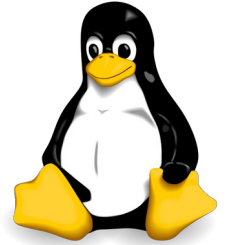
```
RUN apt-get update
```

```
RUN dpkg --add-architecture i386
```

→ Architektur 32-Bit festlegen

```
RUN apt-get install -y cups cups-client  
cups-bsd foomatic-db printer-driver-all  
openprinting-ppds hpijs-ppds hp-ppd
```

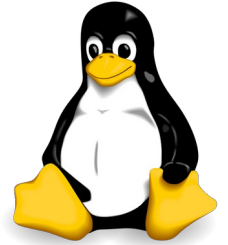
→ CUPS installieren



LUG Albtal

Datei Dockerfile:

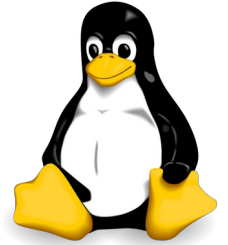
```
RUN apt-get install -y gdebi-core → Pakete für 32-Bit
libxml2:i386 gtk2-engines-murrine:i386
libcanberra-gtk-module:i386 libatk-adaptor:i386
libgail-common:i386
RUN apt-get install -y wget
RUN wget → Acrobat Reader installieren
ftp://ftp.adobe.com/pub/adobe/reader/unix/9.x/9.5.5/
enu/AdbeRdr9.5.5-1_i386linux_enu.deb
RUN dpkg -i AdbeRdr9.5.5-1_i386linux_enu.deb
```



LUG Albtal

Datei Dockerfile:

`CMD service cups start & acroread` → CUPS und Acrobat Reader starten



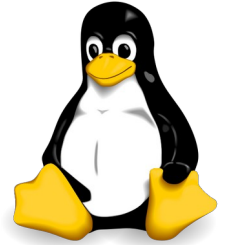
LUG Albtal

Image erstellen

```
cd /acroread  
docker build -t acroread .
```

Ablauf:

1. Basis-Paket wird von Docker hub geladen
2. Pakete werden installiert
3. Fertiges Image wird gespeichert



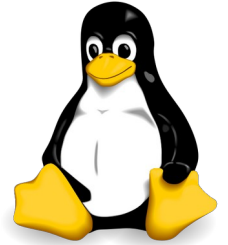
LUG Albtal

Image ausführen

```
docker run --net=host -env="DISPLAY"  
--volume="$HOME/.Xauthority:/root/.Xauthority:rw"  
--volume="/work/daten:/mnt"  
--volume="/work/ppd:/etc/cups/ppd"  
--volume="/work/printers.conf:/etc/cups/printers.conf"  
acroread
```

Auf Host muß vorhanden sein:

- Ordner /work/daten → PDF-Dateien
- Ordner /work/ppd, Datei /work/printers.conf → Drucker für CUPS



LUG Albtal

Image ausführen

1.Container starten: `docker run`

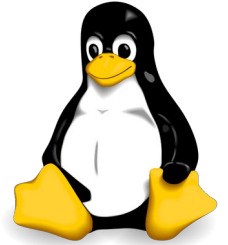
2.Netzwerk mit Gastsystem aufbauen: `--net=host`

3.Lokaler Bildschirm übergeben: `--env="Display"`

4.Laufwerke/Dateien mit Host verbinden: `--volume`

5.CUPS und Acrobat Reader starten:

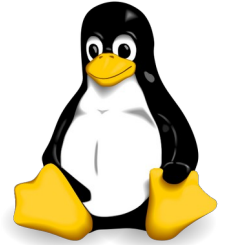
siehe Dockerfile: `CMD service cups start & acroread`



LUG Albtal

Was passiert, wenn das Image gestartet wird:

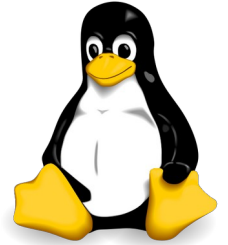
- Container startet jedes Mal neu → lokale Daten gehen verloren
- Container kann nur über Netzwerk angesprochen werden → verhält sich wie ein eigener Rechner



LUG Albtal

Kommandos:

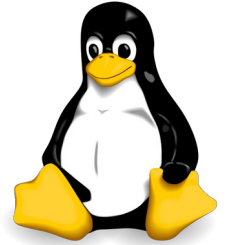
- Info zu aktiven Container: `docker ps`
- Info zu Docker: `docker info`
- Shell im Container starten:
`docker exec -ti <Container> /bin/bash`
- Image löschen: `docker rm -t <Image>`



LUG Albtal

Weitere Informationen und Quellen:

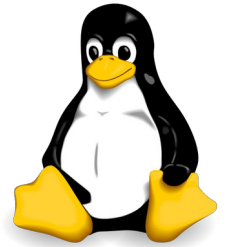
- [https://de.wikipedia.org/wiki/Docker_\(Software\)](https://de.wikipedia.org/wiki/Docker_(Software))
- <https://www.informatik-aktuell.de/entwicklung/methoden/containerplattform-ego-fuer-devops.html>
- <https://entwickler.de/online/development/docker-basics-system-level-virtualisierung-125514.html>
- <https://www.linuxwiki.de/chroot>
- <https://www.pro-linux.de/artikel/2/1464/ressourcen-verwaltung-mit-control-groups-cgroups.html>
- https://en.wikipedia.org/wiki/Linux_namespaces
- <https://de.wikipedia.org/wiki/UnionFS>



LUG Albtal

Persönliches Fazit:

- Containervirtualisierung ersetzt nicht Virtualisierung mit Hypervisor.
- ... ist performant.
- ... ist ideal für Server-Dienste.
- ... ist ideal für Softwareentwicklung.
- ... setzt Linux-Systemkenntnisse voraus.



LUG Albtal

