Cloud Computing

Kapitel 3: Virtualisierung

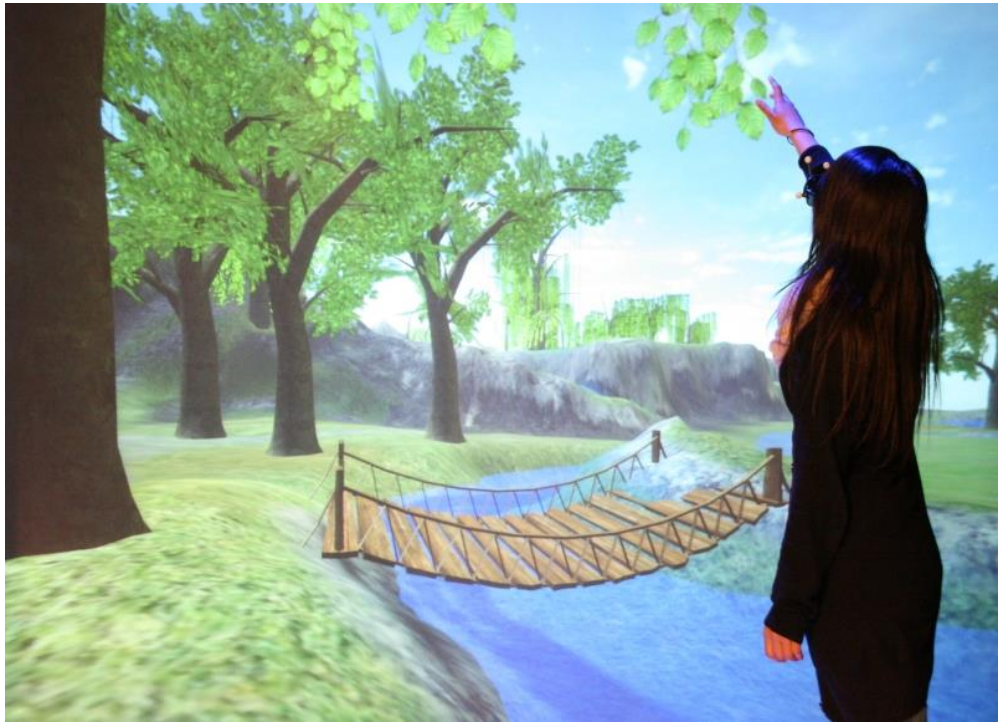
Dr. Josef Adersberger

Grundlagen zur Virtualisierung

Virtualisierung

- **Virtualisierung**: die Erzeugung von virtuellen Realitäten und deren Abbildung auf die physikalische Realität.

- <u>Zweck:</u>
  - **Multiplizität** ➔ Erzeugung mehrerer virtueller Realitäten innerhalb einer physikalischen Realität
  - **Entkopplung** ➔ Bindung und Abhängigkeit zur Realität auflösen
  - **Isolation** ➔ Physikalische Seiteneffekte zwischen den virtuellen Realitäten vermeiden



http://www.techfak.uni-bielefeld.de

Virtualisierungsarten

Virtualisierung ist stellvertretend für mehrere grundsätzlich verschiedene Konzepte und Technologien:

- Virtualisierung von Hardware-Infrastruktur
  1. Emulation
  2. Voll-Virtualisierung (Typ-2 Virtualisierung)
  3. Para-Virtualisierung (Typ-1 Virtualisierung)
- Virtualisierung von Software-Infrastruktur
  4. Betriebssystem-Virtualisierung (*Containerization*)
  5. Anwendungs-Virtualisierung (*Runtime*)

Virtualisierung und Cloud Computing

- Entkopplung von der Hardware für mehr Flexibilität im Betrieb und Robustheit bei Ausfällen.
- Normierung von Ressourcen-Kapazitäten auf heterogener und wechselnder Hardware („S-Instanz“, „XL-Instanz“).
- Zentrale Steuerung und Bereitstellung von Rechen-Ressourcen über die mit Virtualisierung bereitgestellte Software-Defined-Resources.

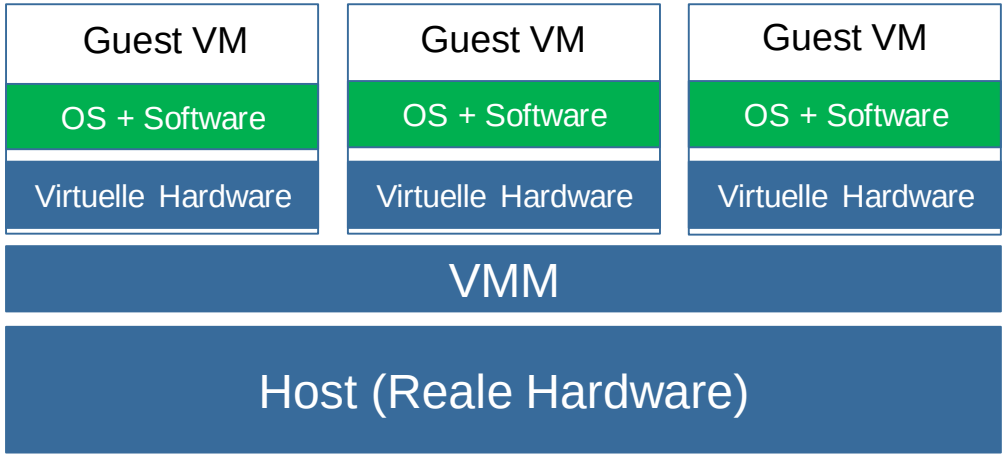
Virtualisierungsarten

Was wird virtualisiert?

- **Prozessor**
  - Virtuelle Rechenkerne
  - Dispatching von Prozessor-Befehlen auf echte Rechenkerne
- **Hauptspeicher**
  - Virtuelle Hauptspeicher-Partition
  - Management der realen Repräsentation (im RAM, auf Festplatte, Balooning)
- **Netzwerk**
  - Virtuelle Netzwerkschnittstellen und virtuelle Netzwerk-Infrastrukturen (VLAN)
  - Brücken zwischen virtuellen und realen Netzwerken
- **Storage**
  - Virtuelle Festplatten-Laufwerke. Abbildung auf Dateien im realen Dateisystem. Volumen entweder vor-allokiert oder dynamisch wachsend.
  - Virtuelle SANs (Storage Area Networks) über Aufteilung der Daten eines virtuellen Laufwerks auf viele Storage-Einheiten.

Hardware-Virtualisierung

- Durch Hardware-Virtualisierung werden die Ressourcen eines Rechnersystems aufgeteilt und von mehreren unabhängigen Betriebssystem-Instanzen genutzt.
- Anforderungen der Betriebssystem-Instanzen werden von der Virtualisierungssoftware (Virtual Machine Monitor, VMM) abgefangen und auf die real vorhandene Hardware umgesetzt.



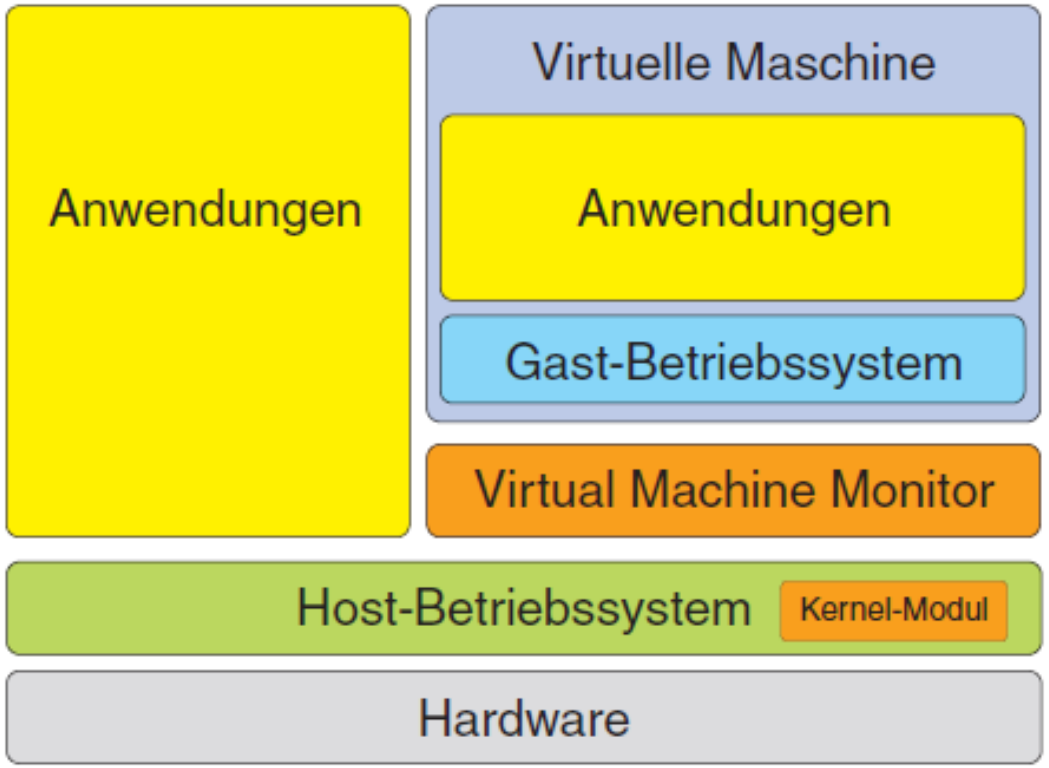
Host
- Der Rechner der eine oder mehrere virtuelle Maschinen ausführt und die dafür notwendigen Hardware-Ressourcen zur Verfügung stellt.

Guest
- Eine lauffähige / laufende virtuelle Maschine

VMM (Virtual Machine Monitor)
- Die Steuerungssoftware zur Verwaltung der Guests und der Host-Ressourcen

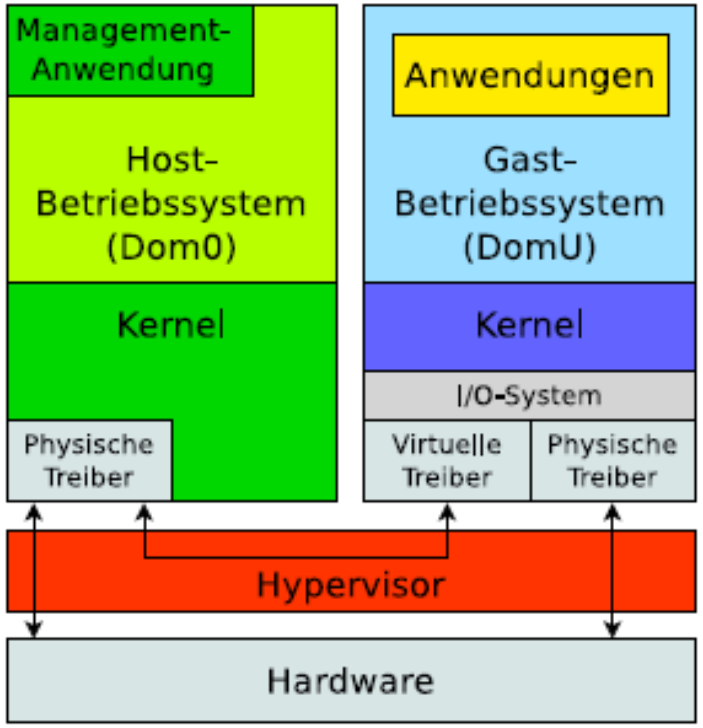
Hardware-Virtualisierung: Voll-Virtualisierung

- Jedem Gastbetriebssystem steht ein eigener virtueller Rechner mit virtuellen Ressourcen wie CPU, Hauptspeicher, Laufwerken, Netzwerkkarten, usw. zur Verfügung
- Der VMM läuft hosted als Anwendung unter dem Host-Betriebssystem (Typ 2 Hypervisor)
- Der VMM verteilt die Hardwareressourcen des Rechners an die VMs
- Teilweise emuliert der VMM Hardware, die nicht für den gleichzeitigen Zugriff mehrerer Betriebssysteme ausgelegt ist (z.B. Netzwerkkarten, Grafikkarten)
- Leistungsverlust: 5-10%.



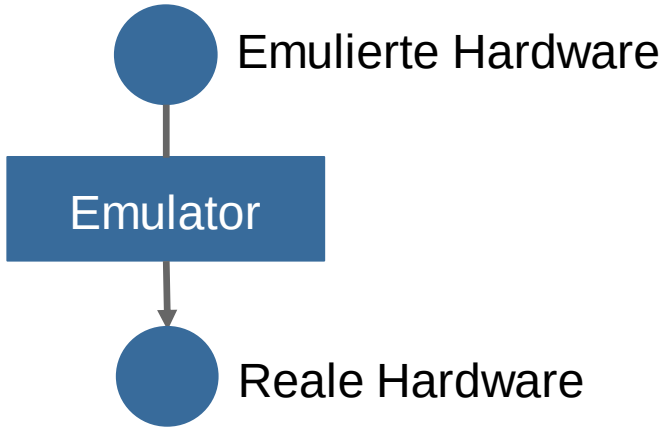
Hardware-Virtualisierung: Para-Virtualisierung

- Der Hypervisor läuft direkt auf der verfügbaren Hardware. Er entspricht somit einem Betriebssystem, das ausschließlich auf Virtualisierung ausgerichtet ist.
- Das Gast-Betriebssystem muss um virtuelle Treiber ergänzt werden, um mit dem Hypervisor interagieren zu können.
  - Dem Gast-Betriebssystem stehen keine direkt low-level virtualisierten Hardware-Ressourcen (CPU, RAM, …) zur Verfügung sondern eine API zur Nutzung durch die virtuellen Treiber.
  - Unterstützte Betriebssysteme und Hardware-Varianten aus Sicht des Gastes eingeschränkt pro Hypervisor-Implementierung.
- Der Hypervisor nutzt die Treiber eines Host-Betriebssystems, um auf die reale Hardware zuzugreifen. Damit brauchen im Hypervisor nicht aufwändig eigene Treiber implementiert werden.
- Leistungsstärkste Virtualisierung (Leistungseinbuße: 2-3%)

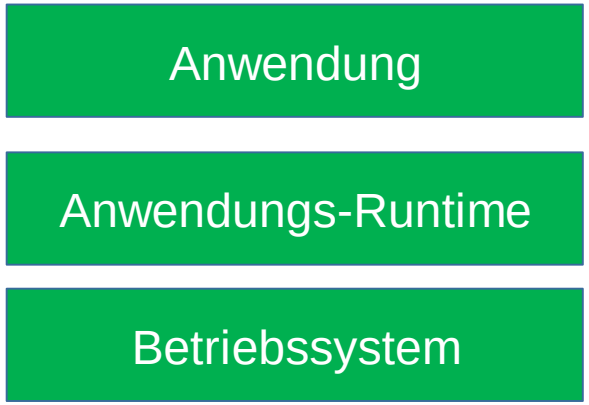


Zur Vollständigkeit: Was ist Emulation und Anwendungs-Virtualisierung?

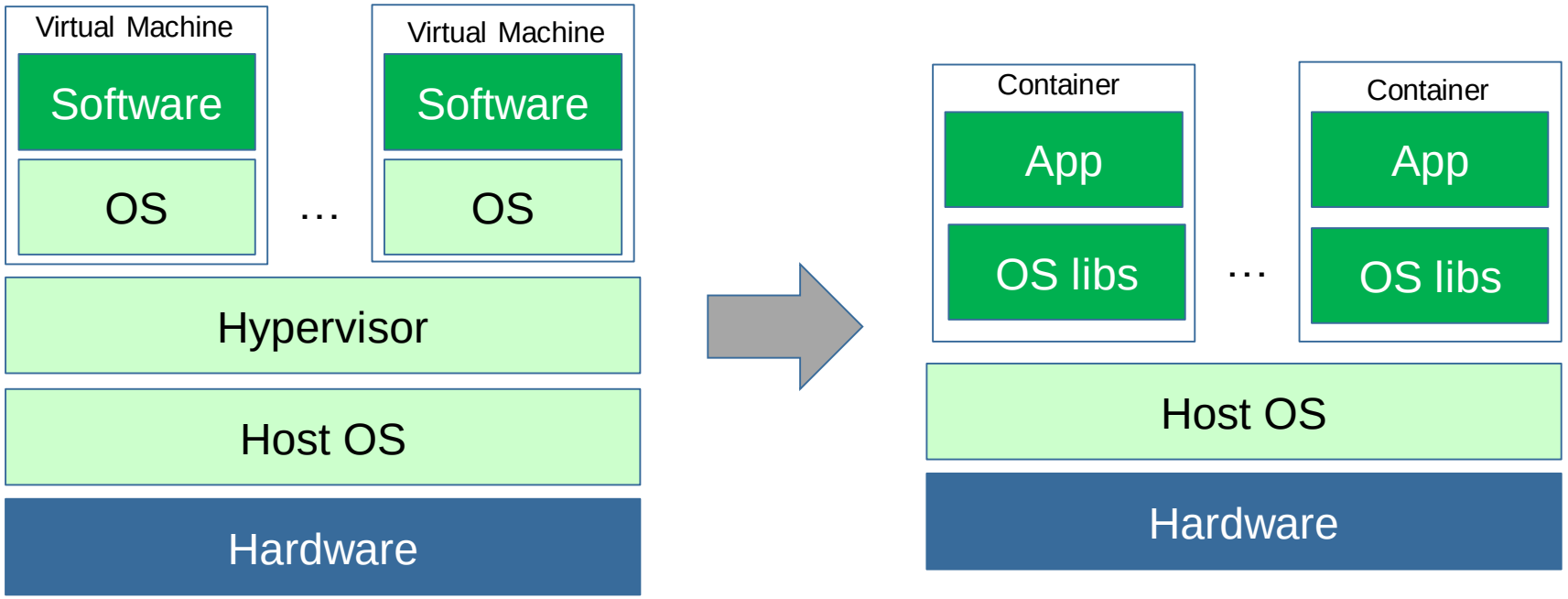
- **Emulation:** Bildet die Hardware eines nicht vorhandenen oder nicht kompatiblen Rechnersystems oder Teile eines entsprechenden Rechnersystems nach. Zweck u.A.: Alte Software konservieren.
  (Beispiel: PearPC)



- **Anwendungs-Virtualisierung:** Stellt Anwendungen eine Programmierschnittstelle und eine Laufzeitumgebung (Runtime) zur Verfügung, die komplett vom darunterliegenden Betriebssystem entkoppelt. Zweck u.A.: Portable Anwendungen.
  (Beispiele: JVM, CLR)



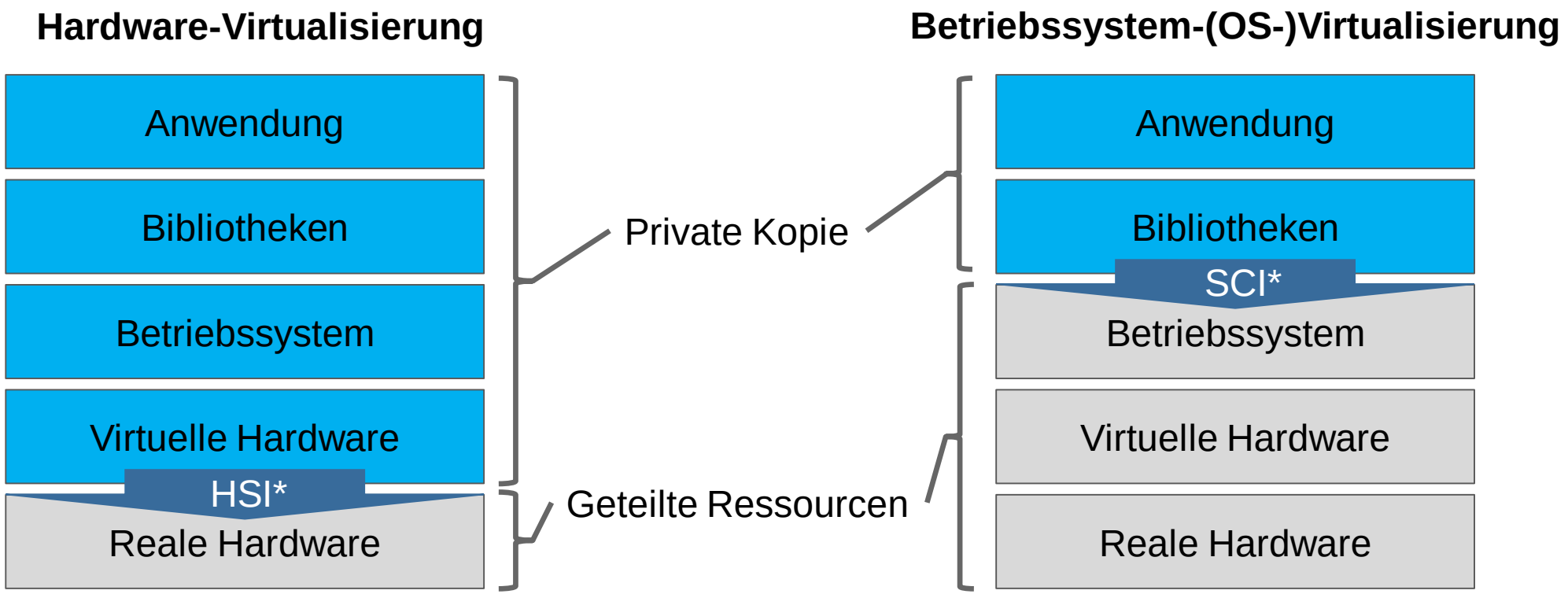
Betriebssystem-Virtualisierung



- Leichtgewichtiger Virtualisierungsansatz: Es gibt keinen Hypervisor. Jede App läuft direkt als Prozess im Host-Betriebssystem. Dieser ist jedoch maximal durch entsprechende OS-Mechanismen isoliert (z.B. Linux LXC).
  - Isolation des Prozesses durch Kernel Namespaces (bzgl. CPU, RAM und Disk I/O) und Containments
  - Isoliertes Dateisystem
  - Eigene Netzwerk-Schnittstelle
- CPU- / RAM-Overhead in der Regel nicht messbar (~ 0%)
- Startup-Zeit = Startdauer für den ersten Prozess

Hardware- vs. Betriebssystem-Virtualisierung

*) HSI = Hardware Software Interface
SCI = System Call Interface

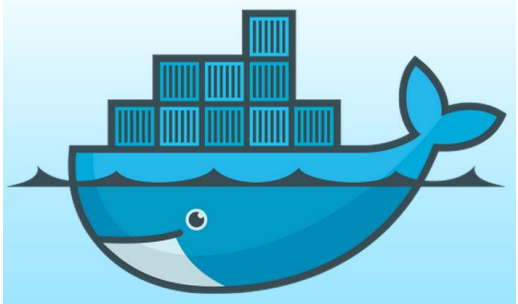


- Stärkere Isolation
- Höhere Sicherheit

- Geringeres Volumen der privaten Kopie
- Geringerer Overhead

Hardware-Virtualisierung: Vagrant und VirtualBox

Betriebssystem-Virtualisierung: Docker



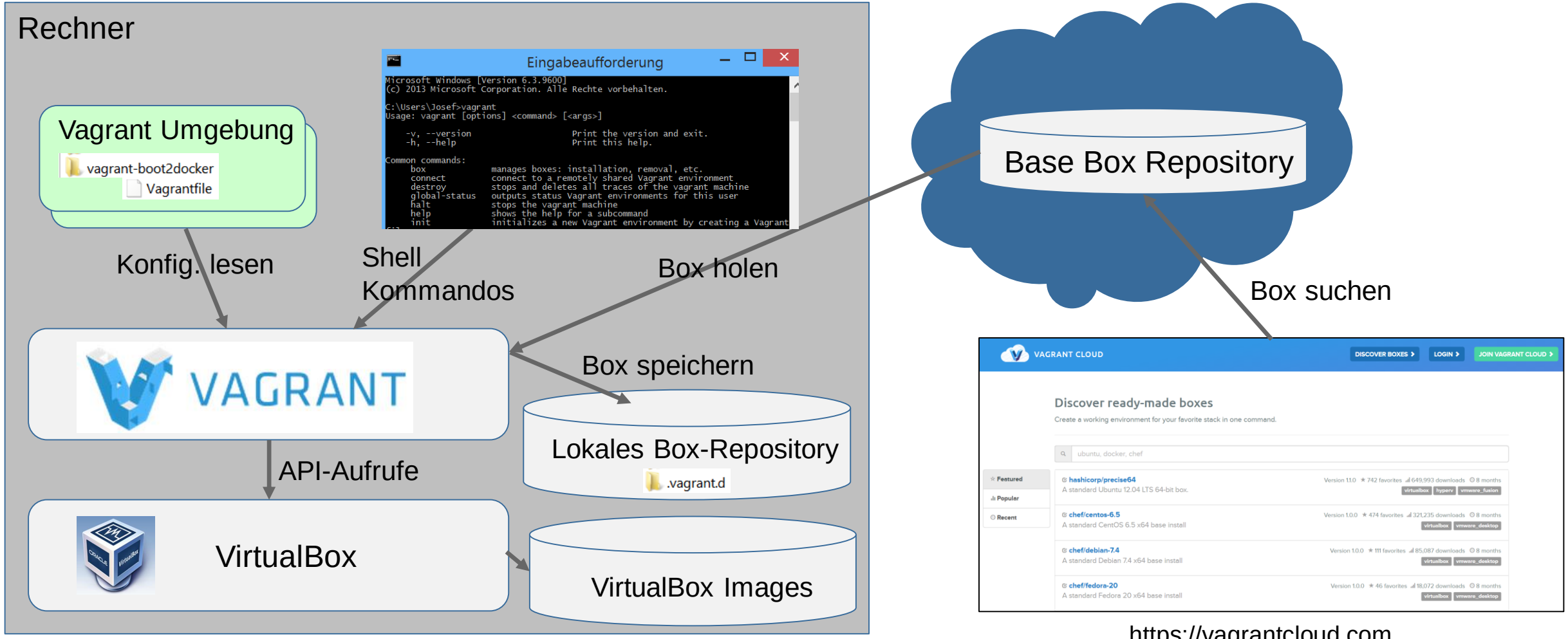
Hardware-Virtualisierung: Vagrant und VirtualBox

+

Open Source Typ 2 Virtualisierungssoftware (Voll-Virtualisierung) für Windows, Linux, OS X und Solaris.

Automationssoftware für virtuelle Umgebungen auf einem Rechner. Virtuelle Maschinen per Kommandozeile erstellen und steuern.

Vagrant: Eine schematische Übersicht.



https://vagrantcloud.com

Das Vagrantfile beschreibt die zu erstellende virtuelle Maschine.

```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
        # My base box
        config.vm.box = "chef/ubuntu-14.04"

        # Define shell provisioning
        config.vm.provision :shell, path: "bootstrap.sh"

        # Define docker provisioning
        config.vm.provision "docker" do |d|
                d.run "nginx1", image: "dockerfile/nginx", args: "-p 8080:80", daemonize: true
                d.run "nginx2", image: "dockerfile/nginx", args: "-p 9080:80", daemonize: true
                d.run "haproxy", image: "dockerfile/haproxy", args: "-p 80:80 --link nginx1:nginx1 --link nginx2:nginx2 -v /vagrant:/haproxy-override"
        end

        # Configure VirtualBox
        config.vm.provider "virtualbox" do |v|
                v.memory = 1024
                v.cpus = 4
        end

        # Forward ports
        config.vm.network :forwarded_port, host: 80, guest: 80
        config.vm.network :forwarded_port, host: 8080, guest: 8080
        config.vm.network :forwarded_port, host: 9080, guest: 9080
end
```

Vagrantfiles werden in Ruby geschrieben

Definition der Basis-Box

Konfiguration der Provisionierung

Konfiguration des Virtualisierungs-Providers

Konfiguration des Netzwerks

Ein typischer Arbeitsablauf mit Vagrant.

#	Befehle auf Kommandozeile	Bedeutung
1	`md <box-dir>` `cd <box-dir>`	Verzeichnis für Vagrant Umgebung erstellen und dorthin wechseln
2	`vagrant init [<box-name>] [<box-url>]`	Eine Vagrant Umgebung initialisieren. Dabei wird zunächst nur eine Datei *Vagrantfile* erstellt und initial mit dem Namen und der URL der Box (falls angegeben) initialisiert.
3		Vagrantfile anpassen nach Bedarf (z.B. IP vergeben, Port-Mapping zwischen Host und Guest, Verzeichnis-Share zwischen Host und Guest, …)
4	`vagrant up`	Startet die virtuelle Maschine (Box ➔ virtuelle Maschine) und konfiguriert sie entsprechend dem Vagrantfile
5	`vagrant ssh`	Per SSH auf die virtuelle Maschine verbinden
6	`exit`	Die SSH Kommandozeile in der virtuellen Maschine verlassen
7	`vagrant halt`	Die virtuelle Maschine stoppen

Weitere nützliche Kommandos:

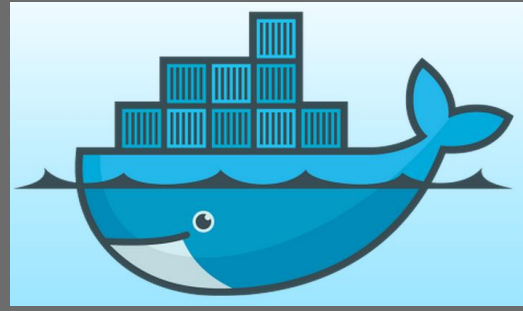
- reload: Startet eine VM neu und aktualisiert die Konfiguration entsprechend dem Vagrantfile
- package: Erstellt aus einer virtuellen Maschine wieder eine Box

Weitere Kommandos: http://docs.vagrantup.com/v2/cli/index.html

Vagrant Befehle auf Kommandozeile

- vagrant box add – allows you to install a box (or VM) to the local machine
- vagrant box remove – removes a box from the local machine
- vagrant box list – lists the locally installed Vagrant boxes
- vagrant init – initializes a project to use Vagrant
- vagrant up – starts up the vagrant VM
- vagrant suspend – saves the state of the current VM.
- vagrant resume – will load up the suspended VM.
- vagrant halt – will shut down the VM, saving configuration. (restart with 'up' command)
- vagrant destroy – will destroy the VM with all config changes.
- vagrant reload – apply Vagrant configuration changes (like port forwarding) without rebuilding the VM.
- vagrant status – tells you the current state of the Vagrant project's VM
- vagrant gem – install Vagrant plugins via RubyGems
- vagrant ssh – short cut to SSH into the running VM
- vagrant package – create a distribution of the VM you have running.
- vagrant <command> –help - Command that will provide man pages for a vagrant command.

Containerization mit Docker



Google Runs All Software In Containers

May 28, 2014 by Timothy Prickett Morgan

The overhead of full-on server virtualization is too much for a lot of hyperscale datacenter operators as well as their peers (some might say rivals) in the supercomputing arena. But the ease of management and resource allocation control that comes from virtualization are hard to resist and this has fomented a third option between bare metal and server virtualization. It is called containerization and Google recently gave a glimpse into how it is using containers at scale on its internal infrastructure as well as on its public cloud.

We are talking about billions of containers being fired up a week here, just so you get a sense of the scale.

http://www.enterprisetech.com/2014/05/28/google-runs-software-containers

Containerization with Docker

http://www.srf.ch/kultur/im-fokus/brasilien/favelas-im-wandel-die-armen-muessen-weichen

Standard format for operations (start, stop, configure, wire, debug, …) and software logistics.

Docker

- Docker ist eine Automationsumgebung für Betriebssystem-Virtualisierung.
- Aktuell unterstützt Docker Linux als Host-Betriebssystem. Eine Windows-Variante ist in Arbeit und erscheint mit Windows Server 2016.
- Docker ist als Werkzeug eines Cloud-Anbieters entstanden und ist mittlerweile eines der sichtbarsten und aktivsten Open-Source-Ökosysteme.

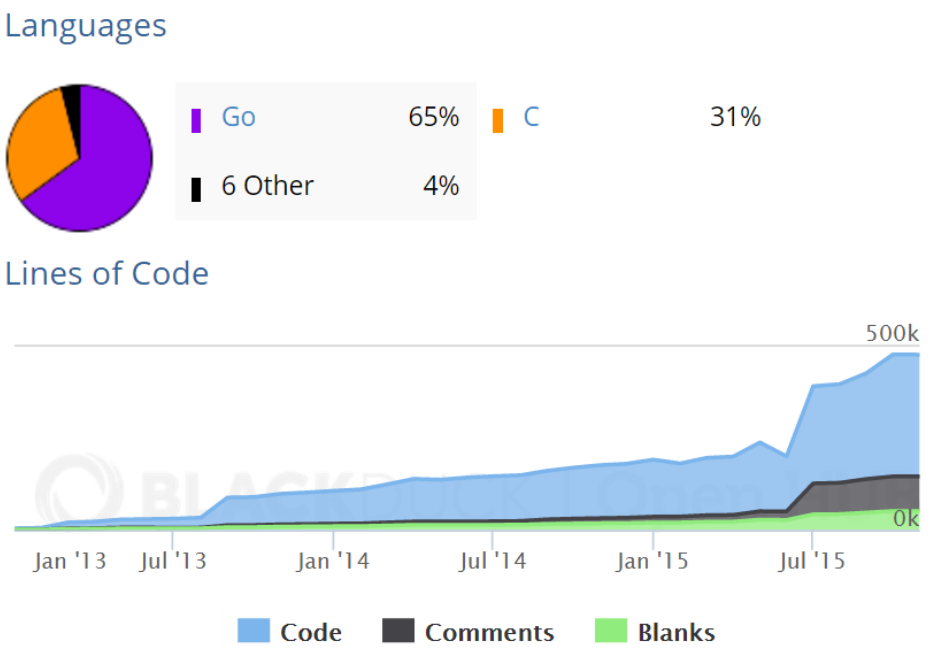
In a Nutshell, docker...

... has had 19,640 commits made by 1,298 contributors
representing 328,622 lines of code

... is mostly written in Go
with an average number of source code comments

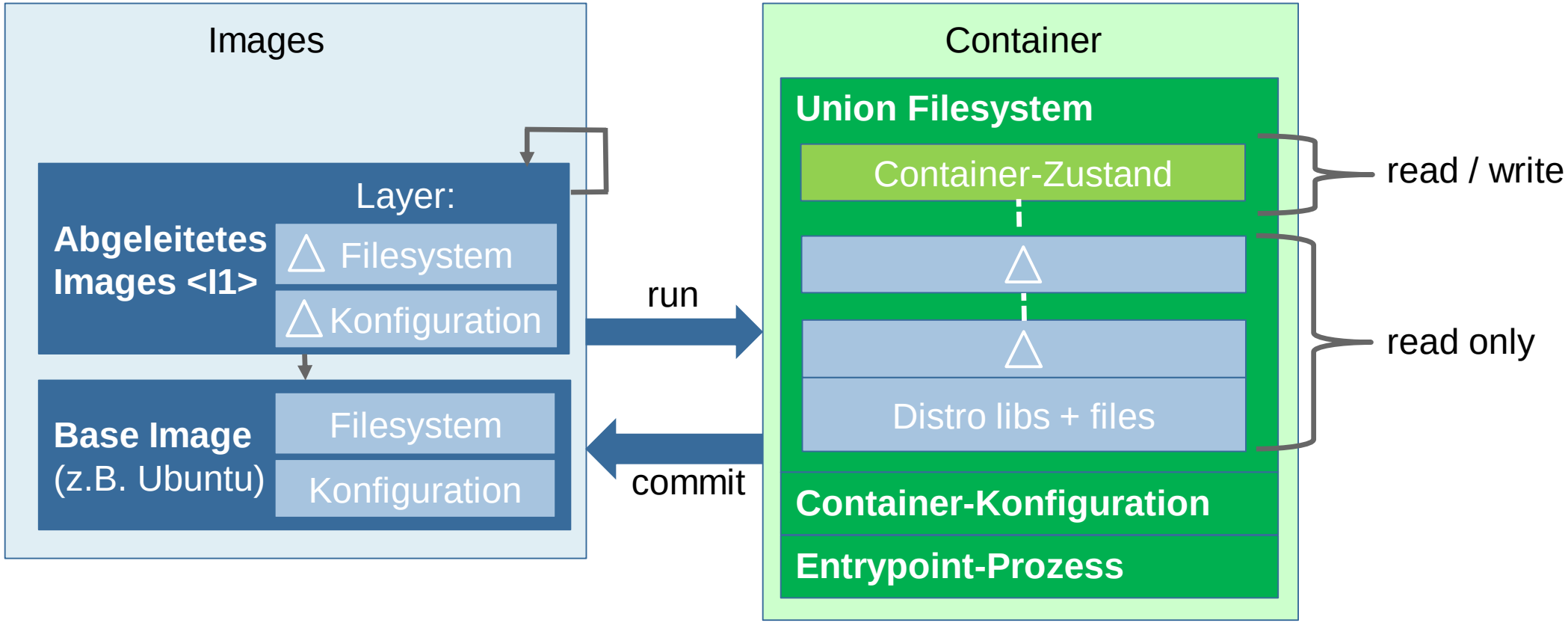
... has a young, but established codebase
maintained by a very large development team
with stable Y-O-Y commits

... took an estimated 88 years of effort (COCOMO model)
starting with its first commit in January, 2013
ending with its most recent commit 3 days ago



https://www.openhub.net/p/docker

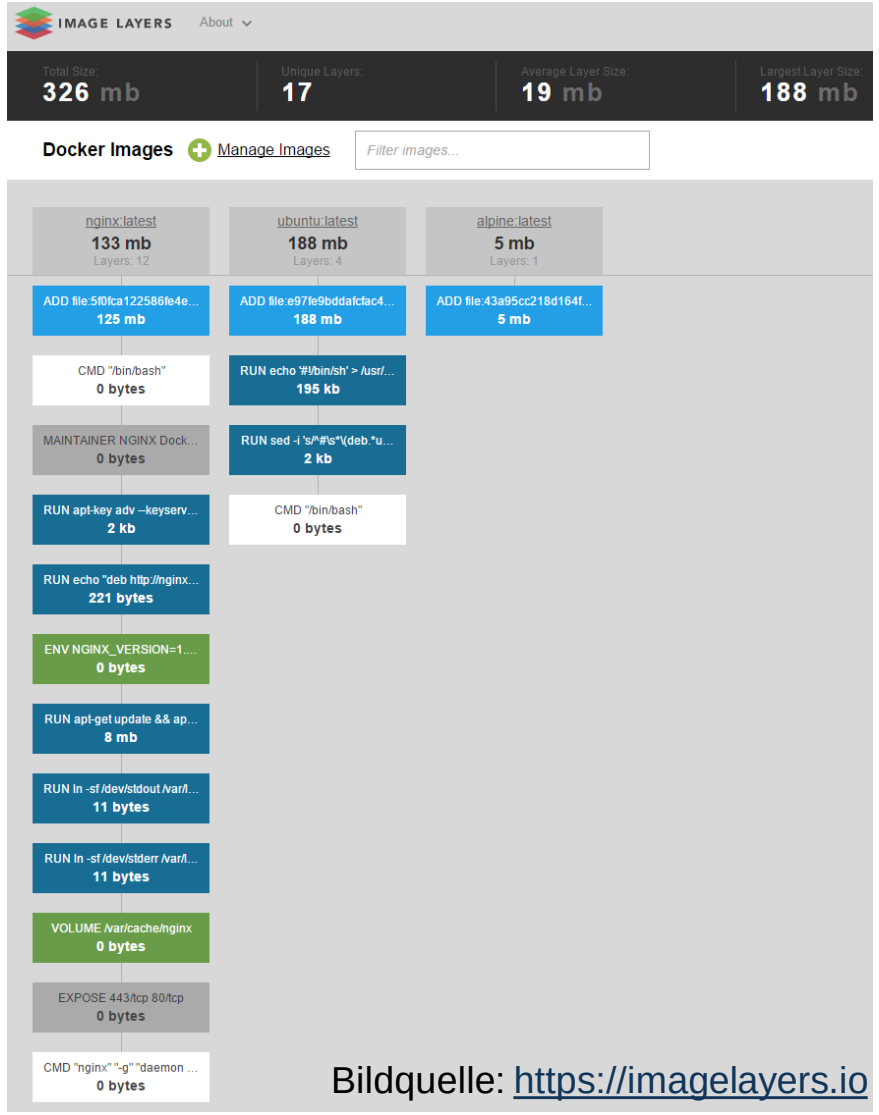
Im Zentrum von Docker stehen Images und Container.



Ruhender und transportierbarer Zustand

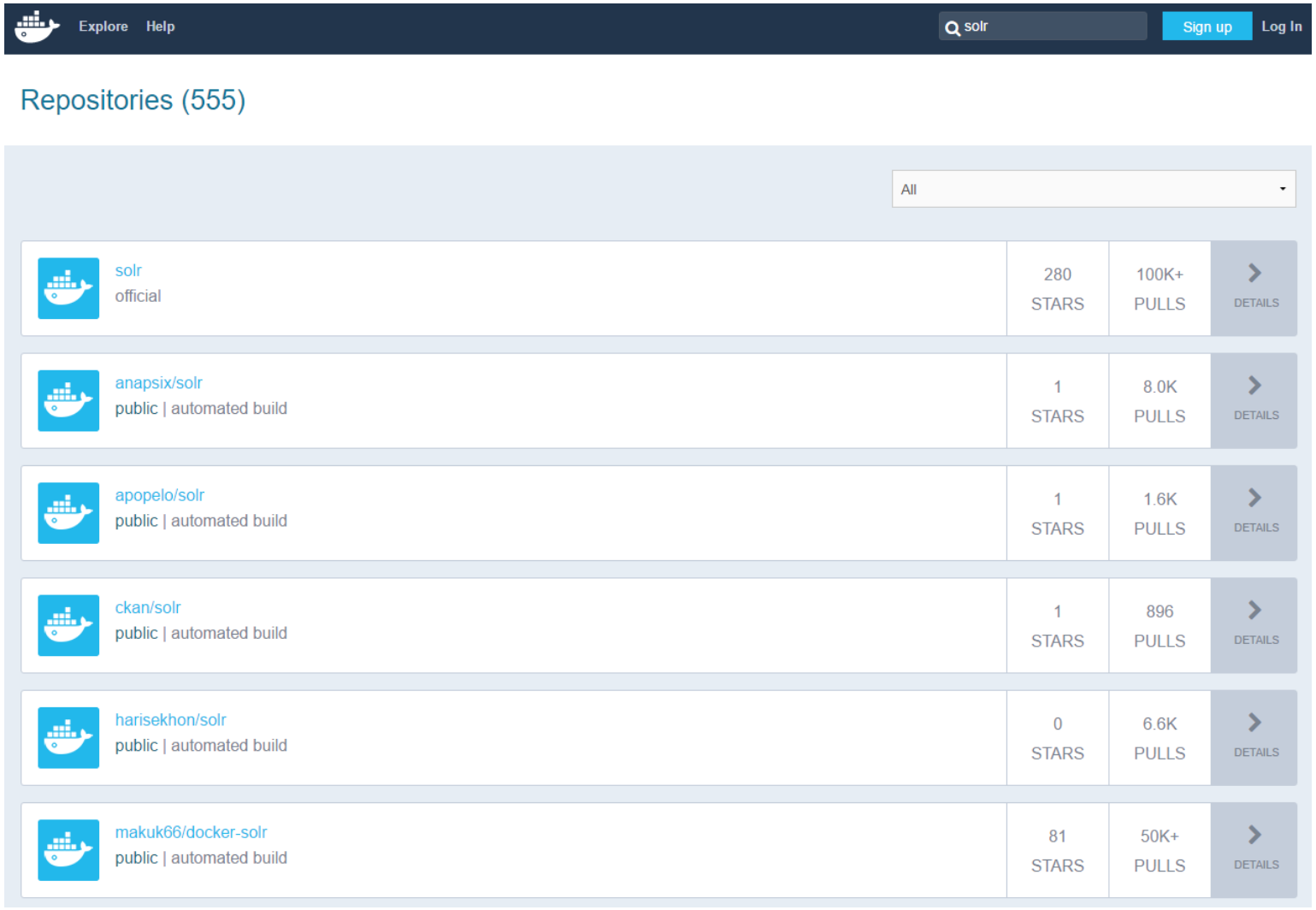
Laufender Zustand
Ein Container läuft so lange wie sein Entrypoint-Prozess im Vordergrund läuft. Docker merkt sich den Container-Zustand.

Visualisierung der Image Layer eines konkreten Images mit dem Werkzeug „Image Layers“.

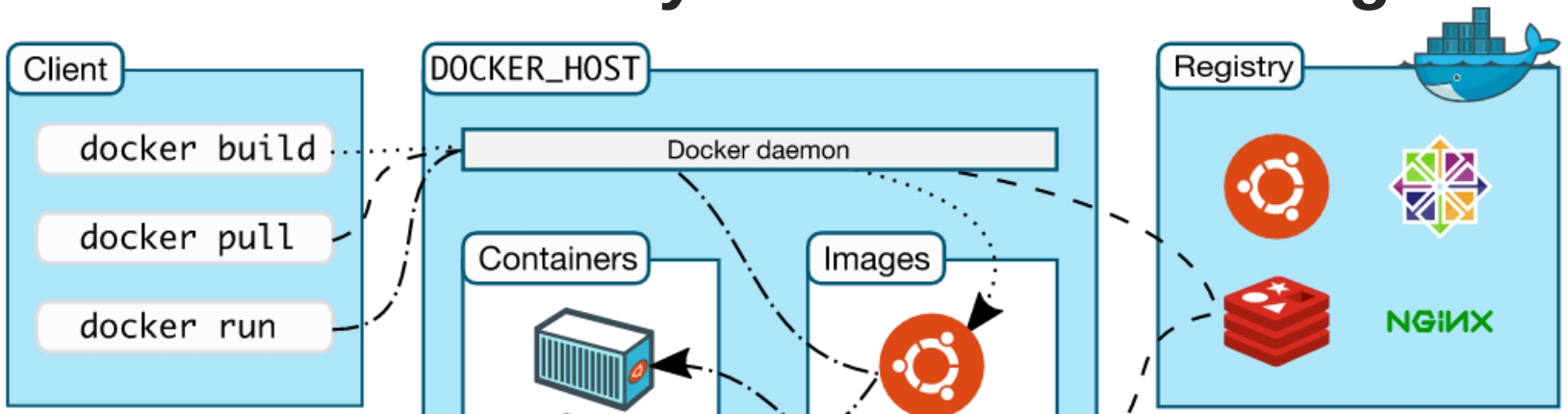


Bildquelle: https://imagelayers.io

hub.docker.com ist die öffentliche Standard-Registry für Docker Images.



Docker ist eine Automationsumgebung für Anwendungs-Container auf Basis Betriebssystem-Virtualisierung.



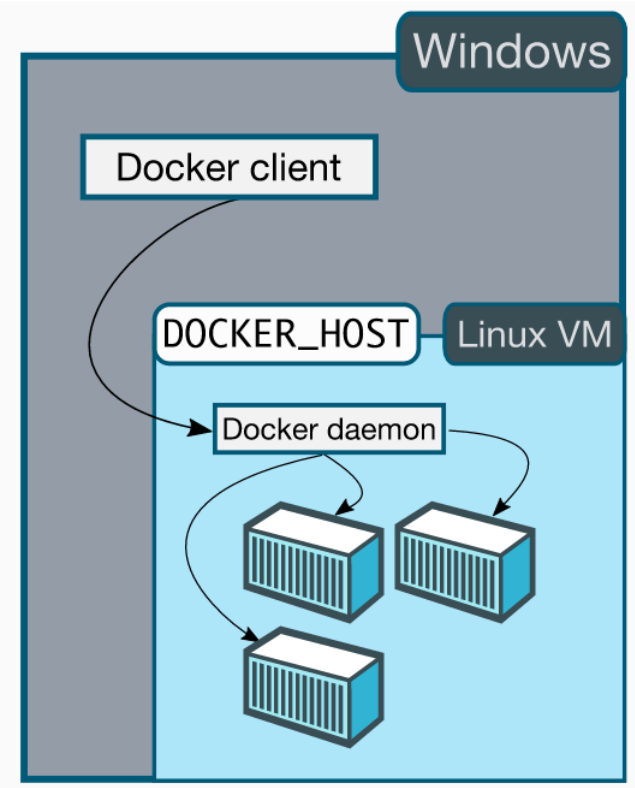
- Kommandozeilen-Werkzeug (Socket-Kommunikation mit Daemon)
- Beliebiger Client (Kommunikation per REST API)

Der Docker Daemon ist die zentrale Steuerungseinheit und läuft direkt als Prozess im Host-Betriebssystem. Er verwaltet alle lokalen Container und Images auf dem Host.

- Öffentliche Registries wie Docker Hub oder Quay.io.
- Unternehmesinterne / private Registries

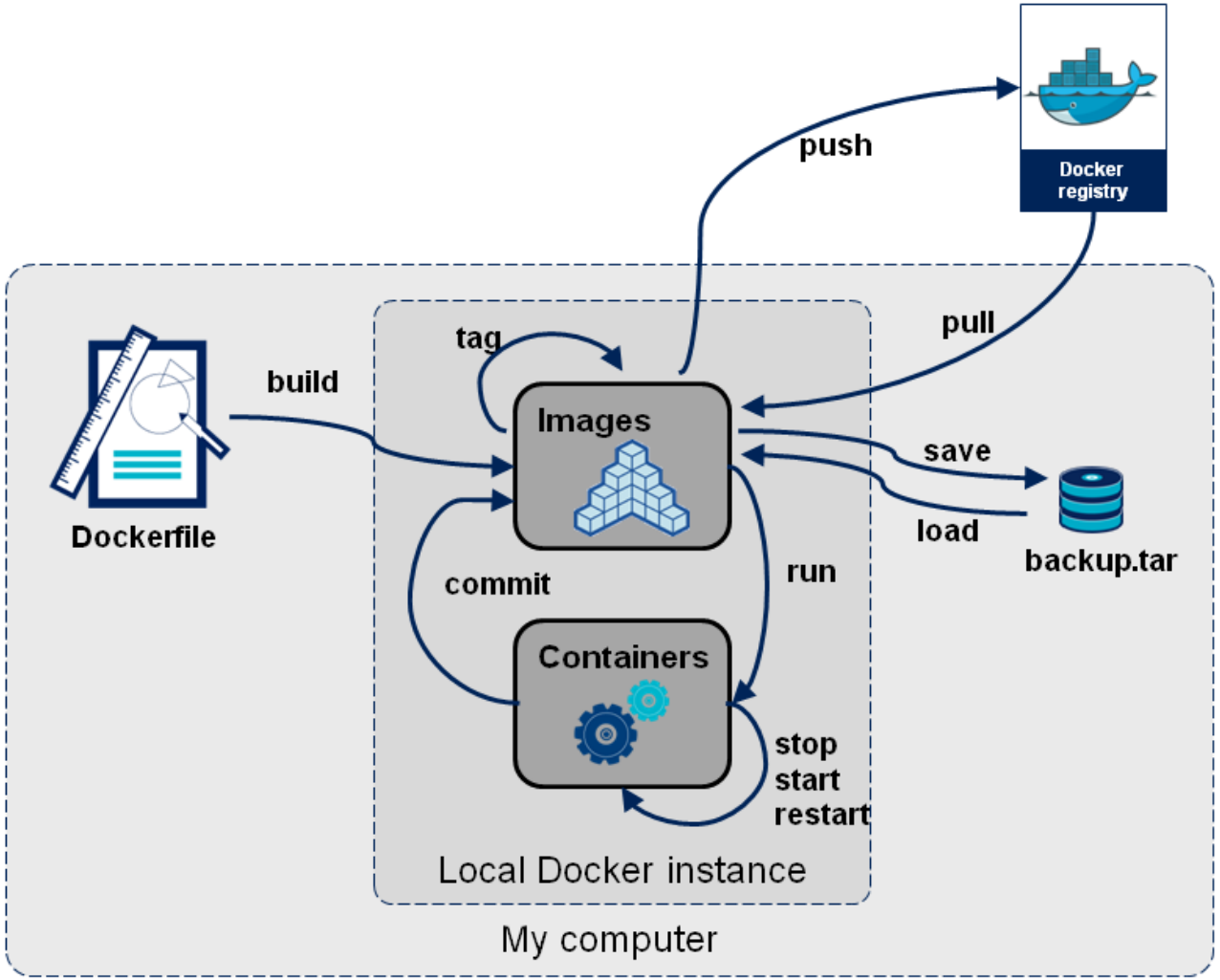
Bildquelle: http://docs.docker.com/engine/introduction/understanding-docker

Beispiel: boot2docker / Docker Machine



Bildquelle: http://docs.docker.com/engine/installation/windows

Das Big Picture von Docker.



A Typical Workflow

Images:
- `busybox:` Mini OS (2MB) for testing purposes
- `alpine:` Mini OS (5MB) with package mgr.
- `ubuntu:` Maxi OS (188MB)

Command	Action
`docker images`	Prints all local images

`docker run`
`-d`
`-v <volume mounts>`
`-p <host-port>:<container-port>`
`<image> <entrypoint process>`
	Run a Docker image: Creates and runs a container.
	▪ in background
	▪ with host directory mounted into the container
	▪ with port forwarding from Host to Container
	▪ image name and entrypoint process

`docker run`
`-ti`
`<image> /bin/sh`
	Run a Docker image and open a shell within the container
	▪ … with forwarding of local terminal
	▪ Image name and shell (or „`/bin/bash`“)

`docker ps -a`	Prints all containers (without –a = only running containers)
`docker commit <container> qaware/foo`	Store container as local image
`docker kill <container>` `docker rm <container>`	Terminate container (send SIGKILL to entrypoint process) Remove container
`docker rmi -f <image>`	Remove local image

More commands: https://coderwall.com/p/2es5jw/docker-cheat-sheet-with-examples, https://docs.docker.com/reference

Container Troubleshooting

Command	Action
`docker inspect <container>`	Shows container metadata (e.g. IP)
`docker logs <container>`	Prints container syslog
`docker top <container>`	Prints all running processes within a container (like `ps -a` within the container)
`docker exec -ti <container> /bin/sh`	Connect terminal to running container
`docker stats`	Shows container runtime statistics (e.g. CPU usage, IO intensity, …)

Ultima ratio: sysdig

Dump system activity to file, so that sysdig can be used to process it later.	`~$ sudo sysdig -w trace.scap`
View the top network connections for a single container.	`~$ sudo sysdig -pc -c topconns container.name=wordpress1`
See the files where apache spends the most time doing I/O.	`~$ sudo sysdig -c topfiles_time proc.name=httpd`
Show all the interactive commands executed inside a given container.	`~$ sudo sysdig -pc -c spy_users container.name=wordpress1`
Show every time a file is opened under /etc.	`~$ sudo sysdig evt.type=open and fd.name contains /etc`

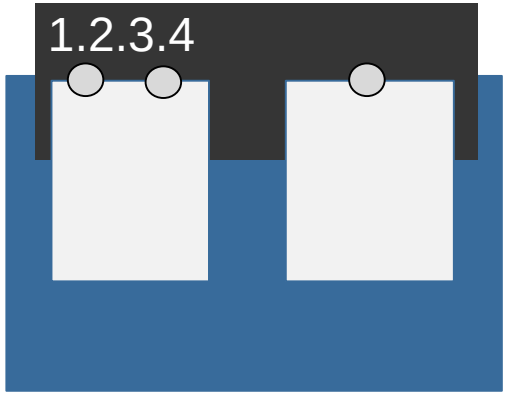
Docker Befehle auf der Kommandozeile

- `docker create` creates a container but does not start it.
- `docker run` creates and starts a container in one operation.
- `docker stop` stops it.
- `docker start` will start it again.
- `docker restart` restarts a container.
- `docker rm` deletes a container.
- `docker kill` sends a SIGKILL to a container.
- `docker attach` will connect to a running container.
- `docker wait` blocks until container stops.

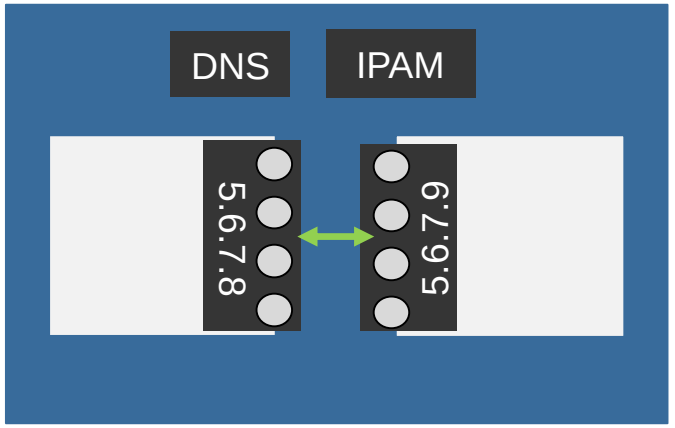
Weitere Kommandos: https://coderwall.com/p/2es5jw/docker-cheat-sheet-with-examples, https://docs.docker.com/reference

Docker Networking Modes

Bridge

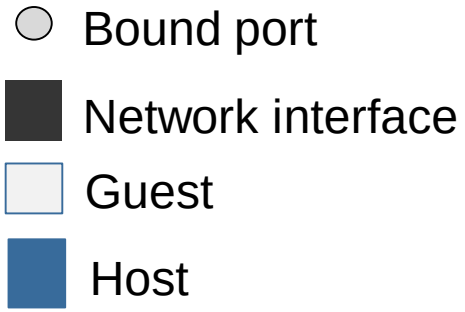


Host

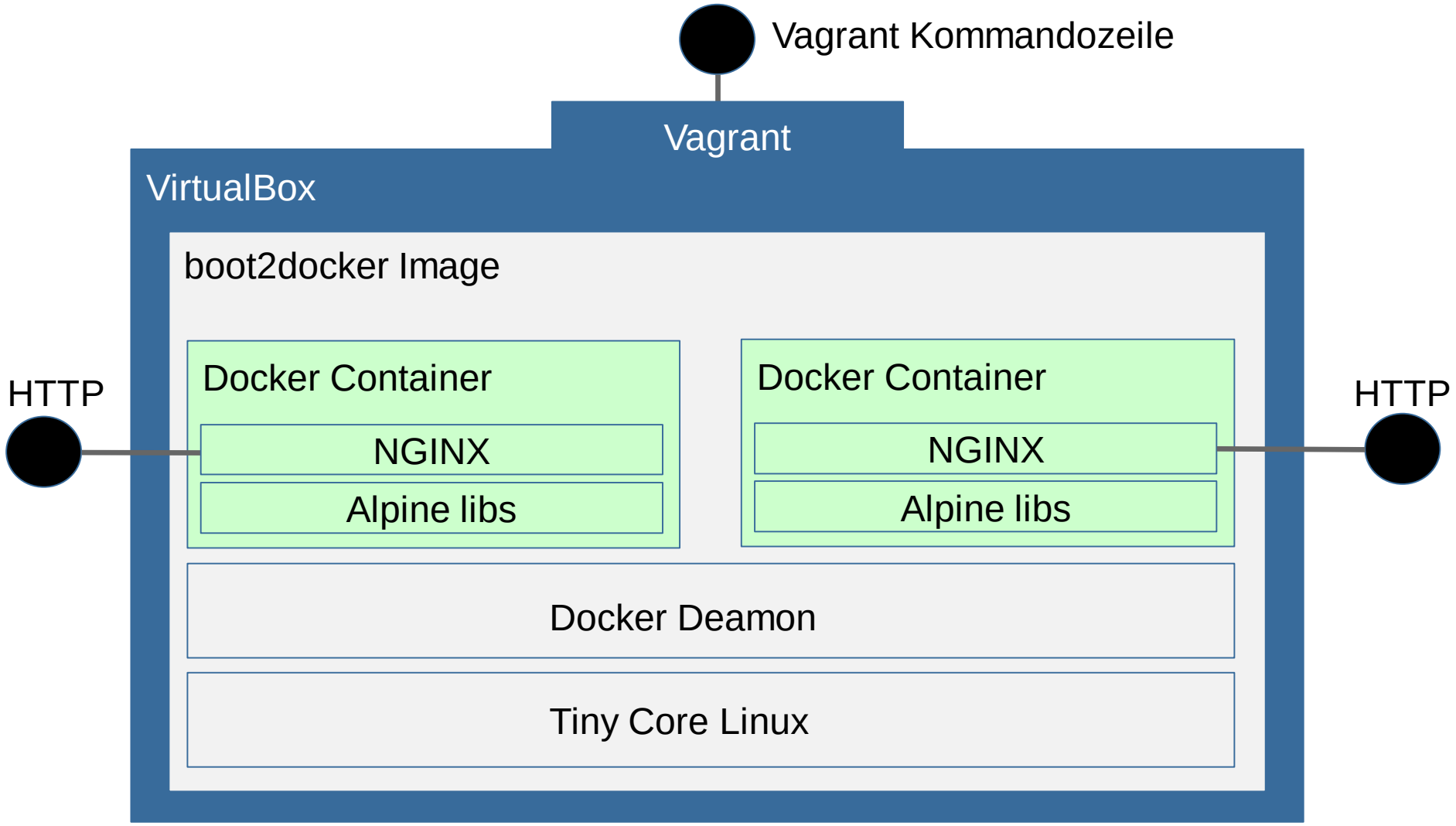


Overlay Network

```
docker network ls
docker network inspect bridge
docker network create --driver overlay multi-host-network
docker network connect multi-host-network container1
```



Die Übung


Vagrant Kommandozeile
Vagrant
VirtualBox
boot2docker Image
HTTP
Docker Container
NGINX
Alpine libs
Docker Container
NGINX
Alpine libs
HTTP
Docker Deamon
Tiny Core Linux

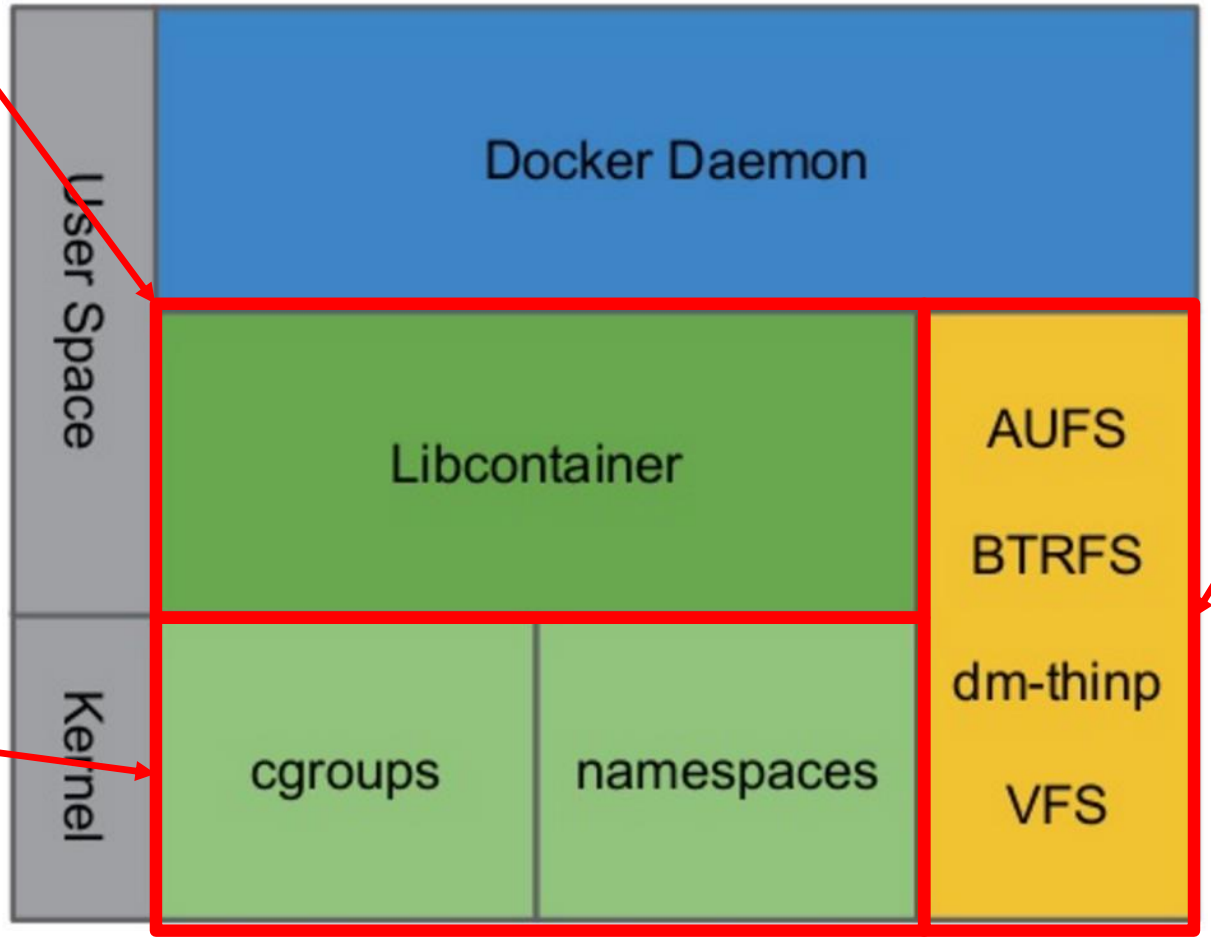
Anhang: Docker von Innen

Docker von Innen: Die Bausteine von Docker

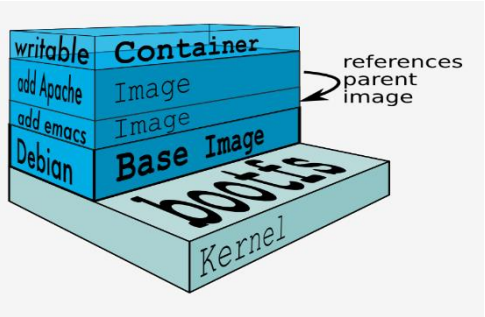
Entkopplung

Aktuell wird neben Libcontainer auch LXC und bald auch das standardisierte Format OCI (https://www.opencontainers.org) unterstützt.

Isolation



Multiplizität



http://docs.docker.com/terms/layer/

Union file systems (UnionFS)

- Ein Union File Systems funktioniert über Layers
- Diese werden zu einem gemeinsamen File System zusammen-gefasst

http://de.slideshare.net/RohitJnagal/docker-internals

Linux Cgroups (Isolation durch Grenzen)

- Ein Feature des Linux-Kernels, das maßgeblich durch Google entwickelt wurde
- Gruppiert Prozesse zu Gemeinschaften mit definiertem und beschränktem Ressourcen-Zugriff auf:
  - Prozessor
  - Hauptspeicher
  - I/O (insb. Netzwerk)
  - Disk
- Die Prozess-Gruppen können geschachtelt sein
- Cgroups stellt dabei für die Prozessgruppen sicher, dass
  - Die Ressourcen limitiert sind und die definierten Grenzen nicht überschritten werden
  - Die aktuell verbrauchten Ressourcen kontinuierlich gemessen und protokolliert werden
  - Dass bei Überschreitung der definierten Grenzen die Prozess-Gruppen eingefroren und neu gestartet werden

Linux Kernel Namespaces (Isolation durch Sichtbarkeit)

- Ein Feature des Linux-Kernels, das die Sicht auf das System einschränkt bzgl.
  - Prozessraum / Prozess-Ids
  - Netzwerk-Schnittstellen
  - Host-Name
  - Dateisystem-Mounts
  - IPC (Inter-Prozess-Kommunikation)
  - Benutzerkonten
- Namespaces können geschachtelt sein

Weiterführende Themen

- Das Docker Ökosystem
- Security-Mechanismen in Docker und Absicherung von Docker (Daemon benötigt aktuell noch root-Rechte!)
- Netzwerk-Themen jenseits des Wirings (z.B. DNS, NAT, …)
- Die unterschiedlichen Docker Filesystem-Backends
- Produktionsreife Docker Container
- Auswahl der passenden Implementierungen (z.B. Filesystem)
- Monitoring von Docker Containern
- Orchestrierung und Verknüpfung von Docker-Containern
- Docker von Innen

https://github.com/veggiemonk/awesome-docker

This repository Search Pull requests Issues Gist

veggiemonk / awesome-docker

Watch 406 Star 4,798 Fork 536

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs

A curated list of Docker resources and projects
http://veggiemonk.github.io/awesome-docker/

801 commits 4 branches 1 release 138 contributors

Create new file Upload files Find file Clone or download

Branch: master New pull request

vegasbrianc committed on GitHub Merge pull request #240 from sixeyed/patch-2 Latest commit 8ff5333 a day ago

.gitignore	gitignore	10 months ago
.travis.yml	Update .travis.yml	28 days ago
CONTRIBUTING....	fix contribution link	6 months ago
LICENSE.md	Create License.md	a year ago
README.md	Added 'Where to start - on Windows' section	3 days ago

README.md

Awesome Docker

A curated list of Docker resources and projects Inspired by @sindresorhus' awesome and improved by these **amazing contributors**.

It's now a GitHub project because it's considerably easier for other people to edit, fix and expand on Docker using GitHub. Just click README.md to submit a pull request. If this list is not complete, you can contribute to make it so.

> **Please**, help organize these resources so that they are *easy to find* and *understand* for new comers. See how to **Contribute** for tips!

If you see a link here that is not (any longer) a good fit, you can fix it by submitting a pull request to improve this file. Thank you!

The creators and maintainers of this list do not receive and should not receive any form of payment to accept a change made by any contributor. The goal of this repo is to index articles, learning materials and projects, not to advertise for profit. **All pull requests are merged by default** and removed if inappropriate or unavailable, or fixed when necessary.

All the links are monitored and tested with awesome_bot made by @dkhamsing

What is Docker ?

> Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud.

Source: What is Docker

Where to start ?

- 10-minute Interactive Tutorial
- Docker Training
- Read this complete article: Basics – Docker, Containers, Hypervisors, CoreOS
- Watch the video: Docker for Developers (54:26) by @jpetazzo
- Docker Jumpstart: a quick introduction
- Docker Curriculum: A comprehensive tutorial for getting started with Docker. Teaches how to use Docker and deploy dockerized apps on AWS with Elastic Beanstalk and Elastic Container Service.
- Install Docker on your machine and play with a few Useful Images
- Try Panamax: Docker Management for Humans It will install a CoreOS VM with VirtualBox and has nice front end
- Install Docker Toolbox Docker Toolbox is an installer to quickly and easily

- Docker Containers on the desktop by @jfrazelle) The **funniest way** to learn about docker! (Tips: checkout her dotfiles and her dockerfiles)
- Container Hacks and Fun Images by @jfrazelle @ DockerCon 2015 **MUST WATCH VIDEO** (38:50)
- Learn Docker Full environment set up, screenshots, step-by-step tutorial and more resources (video, articles, cheat sheets) by @dwyl
- Docker Caveats What You Should Know About Running Docker In Production (written 11 APRIL 2016) **MUST SEE**
- How to Whale Learn Docker in your web browser, no setup or installation required.
- Docker for all - Developers, Testers, DevOps, Product Owners + Videos Docker Training Videos for all

Where to start - on Windows ?

- Windows Containers Quick Start Overview of Windows containers, drilling down to Quick Starts for Windows 10 and Windows Server 2016
- Build And Run Your First Docker Windows Server Container Walkthrough installing Docker on Windows 10, building a Docker image and running a Windows container
- Video: Windows Containers and Docker: The 101 A 20-minute overview, using Docker to run PowerShell, ASP.NET Core and ASP.NET apps
- A Comparative Study of Docker Engine on Windows Server vs Linux Comparing the feature sets and implementations of Docker on Windows and Linux
- Docker with Microsoft SQL 2016 + ASP.NET Demonstration running ASP.NET and SQL Server workloads in Docker
- Running a Legacy ASP.NET App in a Windows Container Steps for Dockerizing a legacy ASP.NET app and runnning as a Windows container
- Exploring ASP.NET Core with Docker in both Linux and Windows Containers Running ASP.NET Core apps in Linux and Windows containers, using Docker for Windows

MENU

Useful Articles

Main Resources

- Docker Weekly Huge resource
- Docker Cheat Sheet by @wsargent **MUST SEE**
- Docker Printable Refcard by @dimonomid
- CenturyLink Labs
- Valuable Docker Links Very complete
- Docker Ecosystem (Mind Map) **MUST SEE**
- Docker Ecosystem (PDF) **MUST SEE** find it on blog by Bryzgalov Peter.
- Blog of @frazelledazzell
- Blog of @jpetazzo
- Blog of @progrium
- Blog of @jwilder
- Blog of @crosbymichael
- Blog of @gliderlabs
- Blog of @sebgoa
- Blog of @codeship
- Digital Ocean Community
- Container42
- Container solutions
- DockerOne Docker Community (in Chinese) by @LiYingJie
- Project Web Dev : (Article series) How to create your own website based on Docker
- Docker vs. VMs? Combining Both for Cloud Portability Nirvana
- Docker Containers on the desktop by @jfrazelle The **funniest way** to learn about docker! (Tips: checkout her dotfiles and her dockerfiles))
- Awesome Linux Container more general about container than this repo, by @Friz-zy.

General Articles

- Getting Started with Docker by @fideloper -- Servers For Hackers is valuable resource. At some point, every programmer finds themselves needing to know their way around a server.
- What is Docker and how do you monitor it?
- How to Use Docker on OS X: The Missing Guide
- Docker for (Java) Developers
- Deploying NGINX with Docker
- Eight Docker Development Patterns
- Rails Development Environment for OS X using Docker
- Logging on Docker: What You Need to Know + see the video (~50min)
- Comparing Five Monitoring Options for Docker
- Minimalistic data-only container for Docker Compose (Written Mar 1, 2015)
- Running Docker Containers with Systemd
- Dockerizing Flask With Compose and Machine - From Localhost to the Cloud -- GitHub Learn how to deploy an application using Docker Compose and Docker Machine (written 17 April 2015)
- Why and How to use Docker for Development (written 28 APR 2015)
- Automating Docker Logging: ElasticSearch, Logstash, Kibana, and Logspout (written 27 APR 2015)
- Docker Host Volume Synchronization (written 1 JUN 2015)
- From Local Development to Remote Deployment with Docker Machine and Compose (written 2 JUL 2015)
- Docker: Build, Ship and Run Any App, Anywhere by Martijn Dwars, Wiebe van Geest, Rik Nijessen, and Rick Wieman from Delft University of Technology (written 2 JUL 2015)
- Joining the Docker Ship Learn how to contribute to docker (written 9 JUL 2015)
- Continuous Deployment with Gradle and Docker Describes a complete pipeline from source to production deploy (includes a complete Spring Boot example project) by @gesellix
- Containerization and the PaaS Cloud -- This article discusses the requirements that arise from having to facilitate applications through distributed multicloud platforms.
- Docker for Development: Common Problems and Solutions by @rdsubhas
- Docker Adoption Data A study by Datadog on the real world Docker usage stastics and deployment patterns.
- How to monitor Docker (4-part series)
- Using Ansible with Docker Machine to Bootstrap Host Nodes by @nathanleclaire
- Swarm v. Fleet v. Kubernetes v. Mesos Comparing different orchestration tools. (written OCT 2015)
- The Shortlist of Docker Hosting There are so many specialized and optimized Docker hosting services available, it's high time for a review to see what's on offer (by Chris Ward).

Portuguese Articles

- Uma rápida introdução ao Docker e instalação no Ubuntu
- O que é uma imagem e o que é um container Docker?
- Criando uma imagem Docker personalizada
- Comandos mais utilizados no Docker

Deep Dive

- Creating containers - Part 1 This is part one of a series of blog posts detailing how docker creates containers. By @crosbymichael
- Data-only container madness

Networking

- Using Docker Machine with Weave 0.10 (written 22 APR 2015)
- How to Route Traffic through a Tor Docker container by @jfrazelle (writtent 20 JUN 2015)
- Demystifing Docker overlay networking. By @nigelpoulton

Metal

- How to use Docker on Full Metal
- CargoOS A bare essential OS for running the Docker Engine on bare metal or Cloud.

Multi-Server

- A Docker based mini-PaaS by @prologic
- A multi-host scalable web services demo using Docker swarm, Docker compose, NGINX, and Blockbridge

Cloud Infrastructure

- Cloud Infrastructure Automation for Docker Nodes

Good Tips

- 24 random docker tips by @csabapalfi
- GUI Apps with Docker by @fgrehm
- Automated Nginx Reverse Proxy for Docker by @jwilder
- Using NSEnter with Boot2Docker
- A Simple Way to Dockerize Applications by @jwilder
- Building good docker images by @jbergknoff
- 10 Things Not To Forget Before Deploying Docker In Production
- Docker CIFS – How to Mount CIFS as a Docker Volume
- Nginx Proxy for Docker (written 9 JUL 2015)
- Dealing with linked containers dependency in docker-compose by @rochacbruno
- Docker Tips by @jmervine
- Docker on Windows behind a firewall by @kaitoedter
- Pulling Git into a Docker image without leaving SSH keys behind by @khash
- 6 Million Ways To Log In Docker by @raychaser
- Dockerfile Generator (ruby script)
- Running Production Hadoop Clusters in Docker Containers
- 10 practical docker tips (Dec 2015) by @josdirksen
- Kubernetes Cheatsheet - A great resource for managing your Kubernetes installation
- Container Best Practices - Red Hat's Project Atomic created a Container Best Practices guide which applies to everything and is updated regurlary.
- Production Meteor and Node Using Docker, Part I by @projectricochet
- Resource Management in Docker by @marekgoldmann

Newsletter

- Docker Team
- CenturyLink Labs
- Tutum
- Shippable
- WebOps weekly

Continuous Integration

- Docker and Phoenix: How to Make Your Continuous Integration More Awesome
- Jenkins 2.0 - Screencast Series by Virendra Bhalothia
- Pushing to ECR Using Jenkins Pipeline Plugin by @mikesir87

Optimizing Images

- Create the smallest possible Docker container
- Creating a Docker image from your code
- Optimizing Docker Images
- How to Optimize Your Dockerfile by @tutumcloud
- Building Docker Images for Static Go Binaries by @kelseyhightower
- Squashing Docker Images by @jwilder
- Dockerfile Golf (or optimizing the Docker build process)
- DockerSlim shrinks fat Docker images creating the smallest possible images.
- SkinnyWhale Skinnywhale helps you make smaller (as in megabytes) Docker containers.
- MicroBadger - Analyze the contents of images and add metadata labels

Service Discovery

- @progrium Service Discovery articles series:
  - Consul Service Discovery with Docker
  - Understanding Modern Service Discovery with Docker
  - Automatic Docker Service Announcement with Registrator

...