



Cloud Computing Virtualisierung

Johannes Ebke 2024-04-11

Cloud Computing lebt von der Skalierung

- Erinnerung NIST-Definition: "Resource Pooling"
- Tendenz seit ca. 20 Jahren: Einzelne Server werden immer leistungsstärker.



Bildquelle: proshop.no

Server in **kube07:** HPE ProLiant DL385 Gen10

2x AMD EPYC 7402 à 24 Kerne, 48 Threads, 180 Watt, 2.8-3.3GHz

⇒ 48 Kerne, 96 Threads

512 GB RAM

2x10GbE Netzwerk

2x1TB SSD via SATA

Nettopreis 2020: 8670€

Stromverbrauch: 200W - 600W(?), 40-120€/Monat

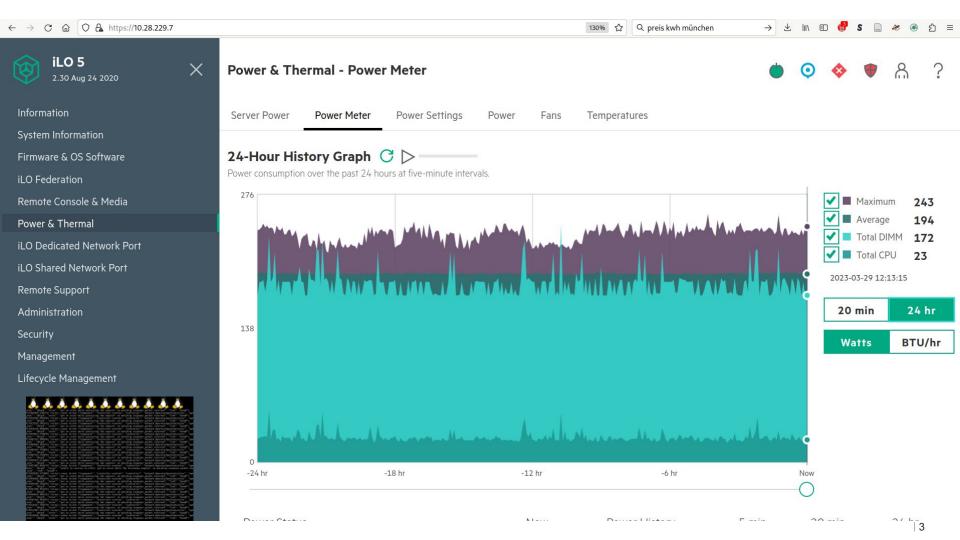
Server auf AWS (via https://instances.vantage.sh/)

r6a.12xlarge

48 vCPUs, 384 GB RAM, 18 Gbit Networking

On-Demand: 2210€/Monat (oder: 5 ct/Minute)

Reserviert: 1461€/Monat Spot Instance: 723€/Monat



Cloud Computing lebt von der Skalierung

- Typischer Use-Case: Datenbank für Wordpress / kleine und mittlere Websites
 - Datenbankgröße: ~100-500MB
 - CPU-Nutzung: ~0.2 CPU
- Dazu Web-Server Apache oder Nginx
 - CPU-Nutzung: ~0.1 CPU
- Redundanz durch Einsatz mehrerer Server?





Cloud Computing lebt von der Skalierung

- Erinnerung: NIST-Definition "Resource Pooling"
- Organisation der Ressourcen in einem Rechenzentrum:
 - Effiziente Nutzung der gegebenen Ressourcen zur Minimierung der Kosten
 - **Isolation** der Ressourcen. Kunden sollen andere nicht sehen und auch nicht von ihnen beeinflusst werden. Seiteneffekte sollten vermieden werden, Security ist Ziel.
 - Entkopplung von der Hardware für mehr Flexibilität im Betrieb und Robustheit bei Ausfällen
 - Ressourcen sollen flexibel vergeben werden. Steuerung mittels "software defined resources"
- · Virtualisierung löst diese Anforderungen und macht Cloud Computing erst möglich.

Virtualisierungsarten

Virtualisierung ist stellvertretend für mehrere grundsätzlich verschiedene Konzepte und Technologien:

- Virtualisierung von Hardware-Infrastruktur
 - Emulation
 - Voll-Virtualisierung
 - Para-Virtualisierung

- Virtualisierung von Software-Infrastruktur
 - Betriebssystem-Virtualisierung (*Containerization*)
 - Anwendungs-Virtualisierung (Runtime)

Virtualisierungsarten: Hardwarevirtualisierung

Was wird virtualisiert?

Hardwarevirtualisierung arbeitet auf Ebene der Rechnerarchitektur.

Prozessor

- Der State des Prozessors. Im wesentlichen Prozessorregister.
- Maschinencode
- Memory Management Unit

Hauptspeicher

· Linear addressierter physikalischer Speicher

Netzwerk

• z. B. Input-Output Stream von Ethernet Frames

Storage

• Blockspeicher (linear addressiert. Lesen und Speichern von Blöcken)

Grafikkarte

- z. B. Framebuffer (2D-Array mit Pixeldaten)
- · 3D Funktionalität (DirectX, OpenGL), siehe https://en.wikipedia.org/wiki/GPU virtualization
- Computing (KI, Simulationen) (Zunehmend wichtig f
 ür die Cloud)
- Evtl. Peripherie wie USB, Maus, Keyboard
- Timer, Interrupt Controller

Was ist Emulation?

- Emulation: Bildet die Hardware eines nicht vorhandenen oder nicht kompatiblen Rechnersystems oder Teile eines entsprechenden Rechnersystems nach
- Emulationen sind sind der Regel sehr langsam und nicht parallelisierbar
- Anwendungen
 - Alte Software konservieren. Zum Beispiel alte Spielekonsolen oder Apollo Guidance Computer: https://svtsim.com/moonjs/agc.html
 - Embedded Entwicklung ohne echte Hardware
 - Reine CPU Emulation
 - Rosetta von Apple. CPU Instruktionsübersetzung von Power-PC zu x86 mittels dynamic binary translation, also Just in Time Übersetzung. (2004). Gerade wird Rosetta 2 entwickelt für die Intel -> ARM Transition
 - https://www.heise.de/news/Windows-on-ARM-Testphase-fuer-x64-Emulation -startet-im-November-2020-4918453.html
 - QEMU User Mode Emulation
 - Voll-Virtualisierer (Hardware und spezielle Instruktionen der CPU)
 - QEMU und Bochs können Windows und Linux praktisch überall starten.



Windows 95 auf der Apple Watch



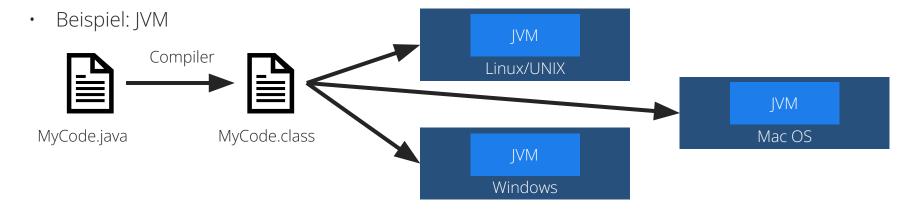
Zur Vollständigkeit: Was ist Anwendungs-Virtualisierung?

 Anwendungs-Virtualisierung: Stellt Anwendungen eine Programmierschnittstelle und eine Laufzeitumgebung (Runtime) zur Verfügung, die komplett vom darunter-liegenden Betriebssystem entkoppelt.
 Zweck u.A.: Portable Anwendungen.

Anwendung

Anwendungs-Runtime

Betriebssystem



Virtualisierung, aber performant

- Emulationen erfüllen zwar viele Voraussetzungen für das Cloud Computing wie Isolation und Entkopplung, sind aber bei der Nachbildung einer ganzen Rechnerarchitektur sehr langsam und daher ungeeignet für den massenhaften produktiven Einsatz.
 - · Hauptverantwortlicher dabei ist die CPU.
- Kann man die selben Ziele mit minimalem zusätzlichen Ressourcenaufwand erreichen?
 - Anwort Ja, aber nur wenn die Gast-Rechnerarchitektur des virtualisierten Systems die gleiche ist wie die Host-Rechnerarchitektur.
 - $x86 \text{ Host} \rightarrow x86 \text{ Guest}$

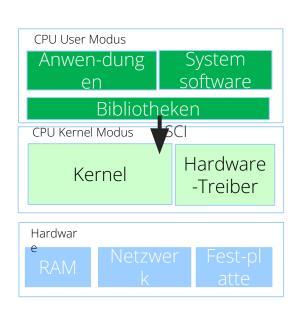
Klassischer Aufbau eines Betriebssystems mit Unterstützung der Rechnerarchitektur

CPU User Mode

- Niedrigste Berechtigungsstufe
 - Keine direkten Hardwarezugriffe
 - Speicherschutz über die Memory Management Unit

CPU Kernel Mode

- User Mode ruft Kernel über das System Call Interface (SCI) auf. Aktuell besteht das SCI bei Linux aus ca. 380 System Calls.
- Höchste Berechtigungsstufe
 - Priviledged CPU Instruktionen
 - Zugriff auf Hardware mittels Treiber
- · Übernimmt z. B. Dateisystemverwaltung und Scheduling der Anwendungen



Unterstützung durch die Hardware

- Software based virtualization
 - In dem klassischen Betriebsmodell hat nur der Usermodus die notwendigen Isolationseigenschaften. Der Kernelmodus kann hier emuliert werden. z. B. mittels "trap-and-emulate". mit wenigstens 10% Performanceverlust.
- Hardware assisted virtualization
 - Oftmals wurden daher CPU Extensions entwickelt wie Intel-VT und AMD-V. Diese fügen einen neuen Prozessormodus (z. B. virtual execution mode) hinzu, bei dem sich das Gastbetriebssystem als mit vollen Privilegien arbeitend wahrnimmt, das Hostbetriebssystem jedoch geschützt bleibt
 - Virtuelle Hauptspeicher-Partition im echten physikalischen Speicher. (Die Null verschiebt sich). Management der realen Repräsentation mittels der Management Memory Unit (MMU).
 - Für die Durchreichung (Pass-Through) der Schnittstellen von echten Hardwaregeräten muss die Verschiebung der Null durch eine IOMMU (I/O Memory Management Unit) ausgeglichen warden.

Hardware-Virtualisierung: Begrifflichkeiten

- Durch Hardware-Virtualisierung werden die Ressourcen eines Rechnersystems aufgeteilt und von mehreren unabhängigen Betriebssystem-Instanzen genutzt.
- Anforderungen der Gastinstanzen werden von der Virtualisierungs-software (Virtual Machine Monitor, VMM) abgefangen und auf die real vorhandene Hardware umgesetzt.
- Der VMM (oder Hypervisor)verteilt die Hardwareressourcen des Rechners an die VMs
- Es werden aber 2 Virtualisierungsmodi und 2 Arten von Hypervisor unterschieden



Host

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

Guest

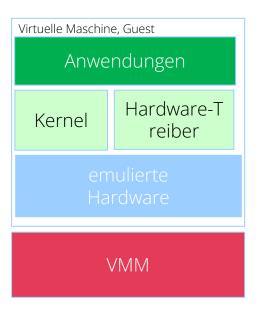
Eine lauffähige / laufende virtuelle Maschine

VMM (Virtual Machine Monitor, auch Hypervisor genannt)

 Die Steuerungssoftware zur Verwaltung der Guests und der Host-Ressourcen

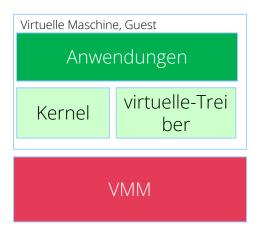
Voll-Virtualisierung

- Jedem Gastbetriebssystem steht ein eigener virtueller Rechner mit virtuellen Ressourcen wie CPU, Hauptspeicher, Laufwerken, Netzwerkkarten, usw. zur Verfügung.
- Das Gastbetriebssystem muss also nicht angepasst werden. Zum Zeitpunkt des Starts muss das Gastbetriebssystem nicht bekannt sein.
- Die VMM emuliert auch weiterhin echte Hardware wie Storage (SATA) und Netzwerk (Ethernet).
- Die VMM kann aber zur Beschleunigung oder zur besseren Nutzung (Grafik, Mouse) spezielle virtuelle Hardware zur Verfügung stellen.
 - z. B. Einfacher Pass-Through von USB
 - Fließender Übergang zur Paravirtualisierung
- Leistungsverlust: 1 5%



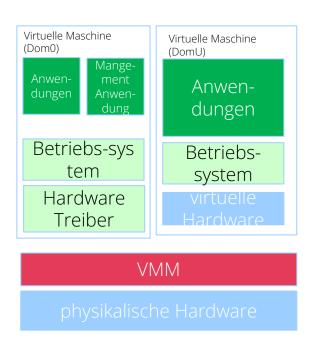
Para-Virtualisierung

- Dem Gast-Betriebssystem stehen keine direkt low-level virtualisierten Hardware-Ressourcen zur Verfügung sondern eine API.
 - Vereinfacht den Aufbau der VM
- Das Gast-Betriebssystem muss portiert werden.
 - Low Level Prozessorinstruktionen werden erst gar nicht ausgeführt oder durch API Aufrufe abgebildet
 - Virtuelle Treiber (z. B. virtio).
 - Vermeidung von Umformungen und Kopieraktionen durch Verwendung spezieller Treiber
 - Übertragung von IP Paketen und nicht von Ethernet Frames.
- Unterstützte Betriebssysteme und Hardware-Varianten aus Sicht des Gastes eingeschränkt pro Hypervisor-Implementierung.
- Leistungsverlust: 0 2%



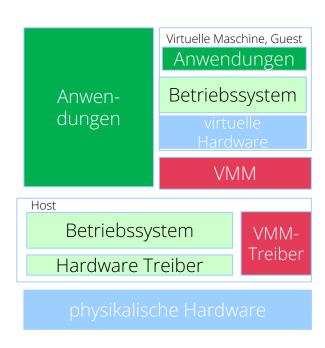
Typ 1: Bare-Metal Virtualisierung

- Der Hypervisor läuft direkt auf der verfügbaren Hardware. Er entspricht somit einem Betriebssystem, das ausschließlich auf Virtualisierung ausgerichtet ist.
- Der Hypervisor nutzt üblicherweise die Treiber eines Host-Betriebssystems, um auf die reale Hardware zuzugreifen. Damit brauchen im Hypervisor nicht aufwändig eigene Treiber implementiert werden.
- Vornehmlich mit Paravirtualisierung, da hier ebenfalls die Emulation der Hardware entfällt
- Ermöglicht einfacheren Pass-Through von echter Hardware. z. B. GPUs an einen Gast
- Beispiele: VMWare ESXi, Microsoft Hyper-V, XEN



Typ 2: Host Virtualisierung

- Der VMM läuft hosted als Anwendung unter dem Host-Betriebssystem
- Vornehmlich bei Voll-Virtualisierung verwendet
- Geringere Skalierbarkeit wegen Abhängigkeit zum Host System
- Mehr Overhead als Typ 1
- Beispiele:
 - Virtualbox
 - VMWare Workstation Player
 - Parallels
 - Achtung: Die Unterscheidung zwischen Typ 1 und Typ 2 ist in vielen Fällen nicht immer klar.



Virtualisierung im Enterprise Umfeld

Neben den bisher genannten Vorteilen bieten heutige VM Lösungen noch viele weitere Features

- Ressourcenverwaltung im laufenden Betrieb
 - Memory Ballooning ungenutzten Hauptspeicher aus VMs dynamisch "wiedergewinnen"
 - · Änderung der Anzahl an virtuellen Rechenkernen
 - Änderung der Festplattengröße mit virtuellen SANs (Storage Area Networks)
- Live Migration
 - · Verschieben der laufenden physikalischen Maschine auf eine andere Hardware innerhalb von Millisekunden
 - CPU State
 - Speicher
 - Storage
 - Netzwerk
- (Echten) Zufall zu erzeugen ist in einer VM noch schwieriger als mit realer Hardware (z. B. mittels Maus und Tastatureingaben). Hier bieten die Hypervisors Schnittstellen an um zusätzlichen Zufall zu erhalten.

Hardware-Virtualisierung mit Vagrant und VirtualBox

Hardware-Virtualisierung: Vagrant und VirtualBox



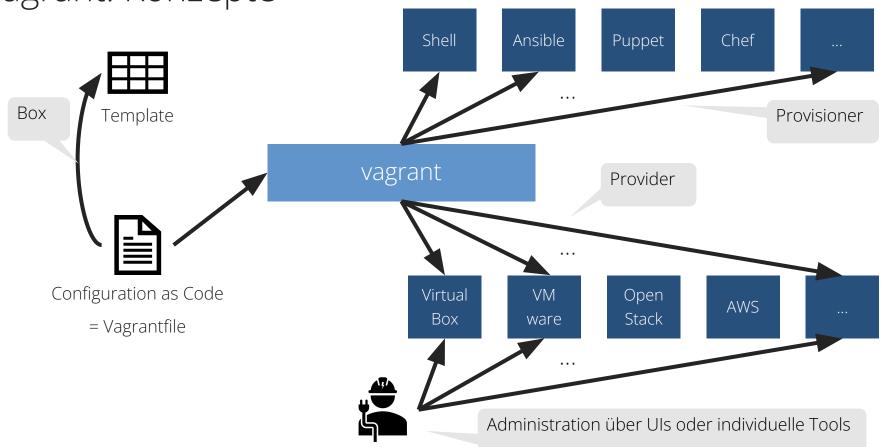




Open Source Typ 2 Virtualisierungs-Software (Voll-Virtualisierung) für Windows, Linux, MacOS und Solaris.

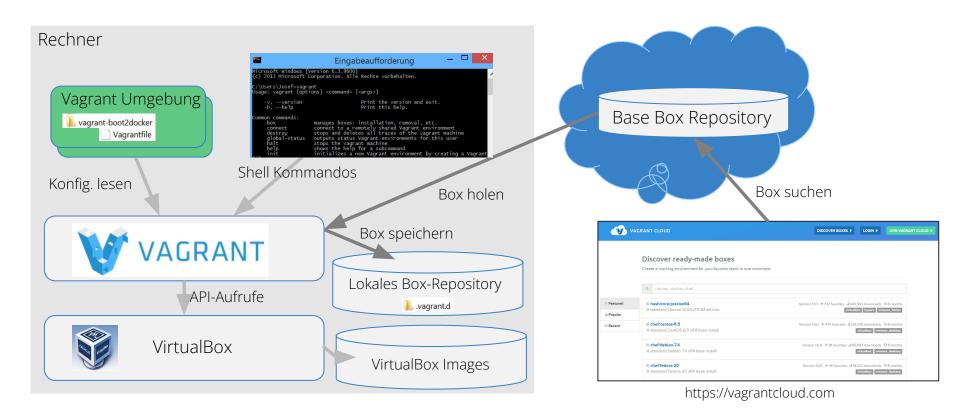
Automationssoftware für virtuelle Umgebungen auf einem Rechner. Virtuelle Maschinen per Kommandozeile erstellen und steuern.

Vagrant: Konzepte





Vagrant: Eine schematische Übersicht.



Das Vagrantfile beschreibt die zu erstellende virtuelle Maschine.

```
Vagrantfiles werden in
# -*- mode: ruby -*-
# vi: set ft=ruby :
                                                                               Ruby geschrieben
# 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
                                                                               Definition der Basis-Box
       config.vm.box = "chef/ubuntu-14.04"
       # Define shell provisioning
       config.vm.provision :shell, path: "bootstrap.sh"
                                                                               Konfiguration der Provisionierung
       # 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|
                                                                               Konfiguration des Virtualisierungs-Providers
               v.memorv = 1024
               v.cpus = 4
       end
       # Forward ports
       config.vm.network :forwarded port, host: 80, guest: 80
                                                                              Konfiguration des Netzwerks
       config.vm.network :forwarded port, host: 8080, guest: 8080
       config.vm.network :forwarded port, host: 9080, guest: 9080
```

Ein typischer Arbeitsablauf mit Vagrant.

#	Befehle auf Kommandozeile	Bedeutung
1	<pre>mkdir <box-dir> cd <box-dir></box-dir></box-dir></pre>	Verzeichnis für Vagrant Umgebung erstellen und dorthin wechseln
2	<pre>vagrant init [<box-name>] [<box-url>]</box-url></box-name></pre>	Eine Vagrant Umgebung initialisieren. Dabei wird zunächst nur eine Datei <i>Vagrantfile</i> 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.
- <u>vagrant reload</u> 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.

Virtualisierungsarten: Betriebssystemvirtualisierung

Hardwarevirtualisierer sind Schwergewichte

- Jede VM inkludiert eine virtuelle Kopie eines kompletten Betriebssystem und benötigt signifikante RAM und CPU Ressourcen, die nur schwer dynamisch geändert werden können
- Softwareentwicklung mit VMs ist träger und komplexer
- Aufgrund der Größe der Images ist die Portabilität ein Problem.
- Kompatibilität mit anderen VM Lösungen nicht vorhanden. Wechsel zwischen Rechenzentren nicht einfach möglich.



Der Urgroßvater: chroot (Jahr 1982)

- chroot gilt als Urgroßvater der Betriebssystemvirtualisierung (Jahr 1982)
- chroot setzt aus Sicht der laufenden Applikation das root Filesystem neu
 - Ermöglicht die Isolation des Filesystems
- Kein Overhead. Implementiert in 2 dutzend Zeilen C-Code im Kernel
- Benötigt root-Rechte
- Keine Netzwerk-Isolation, keine Prozess-Isolation, keine Disk Quotas, keine CPU Quotas, keine I/O Limitierung
- chroot Prozess sieht weiterhin fast alles vom System

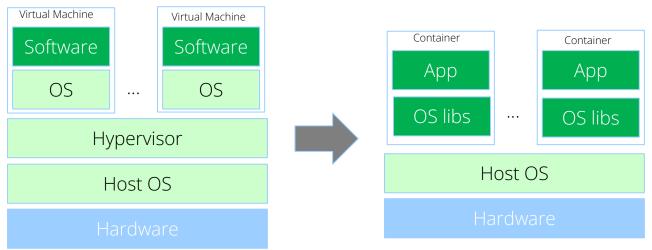
Linux Kernel Namespaces (Jahr 2002)

- Isolation durch Sichtbarkeit
- Ein Feature des Linux-Kernels, das die Sicht und den Zugriff auf das System einschränkt:
 - Prozessraum / Prozess-Ids
 - Netzwerk-Schnittstellen
 - Host-Name
 - Dateisystem-Mounts
 - IPC (Inter-Prozess-Kommunikation)
 - Benutzerkonten
 - 7eit
- Die Einschränkungen sind dabei für den isolierten Prozess transparent.
- Namespaces können geschachtelt sein.
- siehe https://success.docker.com/KBase/Introduction to User Namespaces in Docker Engine

Linux cgroups (Jahr 2007)

- 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 stellen 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
- Siehe https://docs.docker.com/engine/docker-overview/

Betriebssystem-Virtualisierung

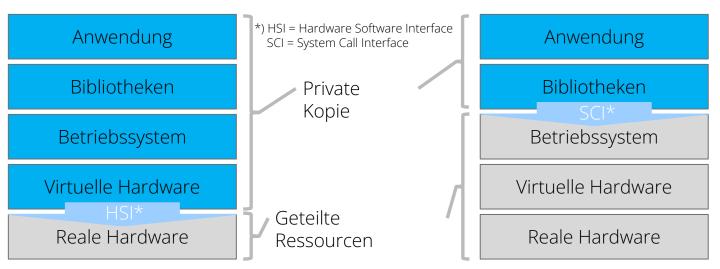


- Leichtgewichtiger Virtualisierungsansatz: Es gibt keinen Hypervisor. Jede App läuft direkt als Prozess im Host-Betriebssystem. Dieser ist jedoch maximal durch entsprechende OS-Mechanismen isoliert (z.B. Linux LXC).
 - Isolation des Prozesses durch Kernel Namespaces (bzgl. CPU, RAM und Disk I/O) und 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

Hardware-Virtualisierung

Betriebssystem-(OS-)Virtualisierung



- Benötigt Hardwareunterstützung
- Höhere Sicherheit. Die HSIs sind einfach.
- Stärkere Isolation.
- Hohes Volumen, Hohe Startzeit
- Unterschiedliche Betriebssysteme

- Ist eine reine Softwarelösung
- Geringere Sicherheit: System Call Interface ist sehr mächtig und komplex
- Geringeres Volumen, Geringerer Overhead, Kürzere Startup-Zeit
- Betriebssystem fest

Containerisierung ist angekommen!

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.

Beispielhafte Technologien

Hardware-Virtualisierung: Vagrant und VirtualBox



Betriebssystem-Virtualisierung: Docker



Betriebssystem-Virtualisierung mit Docker

Containerization mit Docker: Standardisierung



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



Standard format for operations: start, stop, configure, wire, debug + software logistics.

Docker

- Docker ist eine Automationsumgebung für Betriebssystem-Virtualisierung.
- Aktuell unterstützt Docker Linux als Host-Betriebssystem.
 - Seit 2016 steht eine Windows-Variante zur Verfügung, die mit Hyper-V (Typ 1) virtualisierung läuft.
 - Seit 2020 steht mit WSL2 (Windows Subsystem for Linux) auch eine parallel laufende Linux-Kernel zur Verfügung auf dem Docker ausgeführt werden kann.
- Docker ist als Werkzeug eines Cloud-Anbieters entstanden und ist mittlerweile eines der sichtbarsten und aktivsten Open-Source-Ökosysteme. Leider ist Docker (die Firma) inzwischen dazu übergegangen, die freie Nutzung des Dockerhubs einzuschränken - daher kann es Sinn machen, eine eigene Registry zu verwenden.

In a Nutshell, docker...

... has had 228,212 commits made by 5,912 contributors representing 10,917,944 lines of code

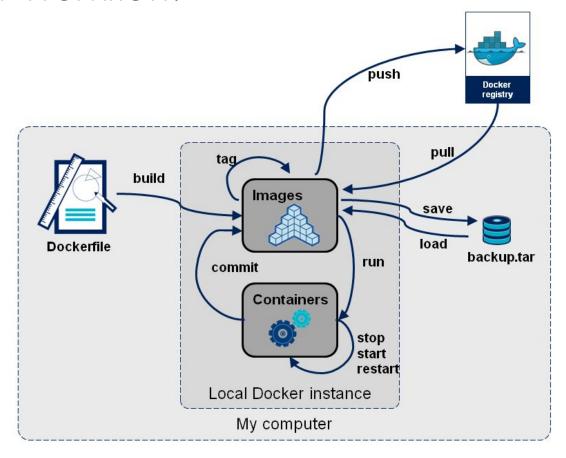
... is mostly written in Go with an average number of source code comments

... has a well established, mature codebase maintained by a very large development team with decreasing Y-O-Y commits

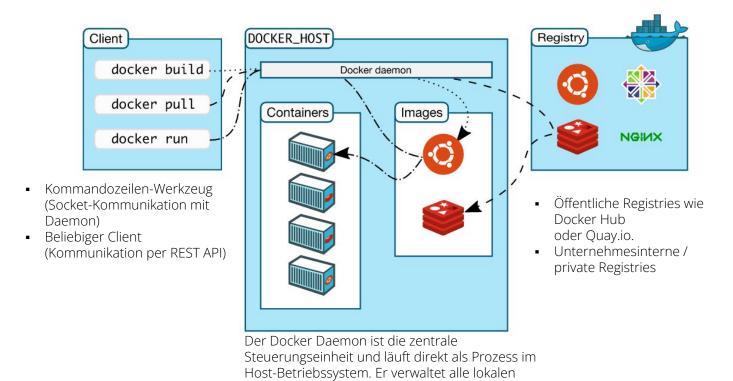
... took an estimated 3,459 years of effort (COCOMO model) starting with its first commit in January, 2012 ending with its most recent commit 4 months ago

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

Der Docker Workflow.

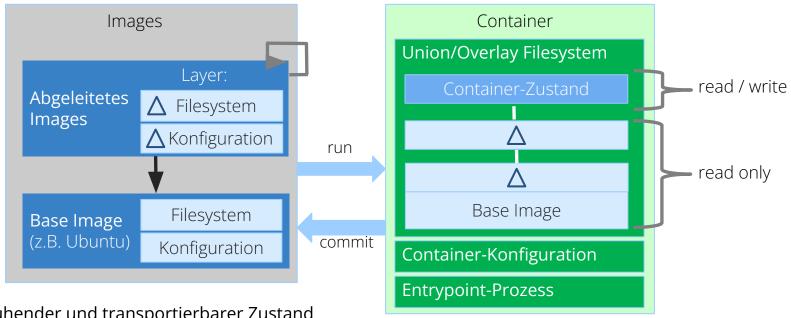


Die Docker Architektur.



Container und Images auf dem Host.

Im Zentrum von Docker stehen Images und Container.



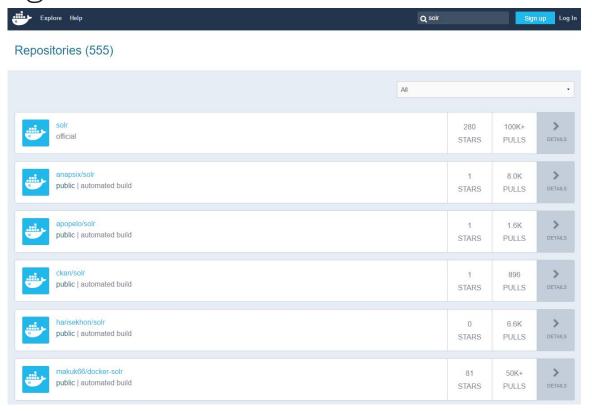
Ruhender und transportierbarer Zustand

- Ein Image basiert i.d.R. auf einem anderen Image und speichert nur das Delta △ zu diesem Image.
- Ausnahme: Das Base-Image

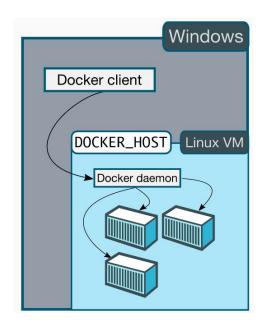
Laufender Zustand

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

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



Zusammenspiel zwischen Host und Docker Machine.



Provisionierung von Images mit dem Dockerfile

Ein Dockerfile erzeugt auf Basis eines anderen Images ein neues Images. Dabei werden die folgenden Aktionen automatisiert:

- Konfiguration des Images und der daraus resultierenden Container
- Ausführung von Provisionierungs-Aktionen

Ein Dockerfile ist somit eine Image-Repräsentation alternativ zu einem physischen Image (Bauanteilung vs. Bauteil).

- Wiederholbarkeit beim Bau von Containern
- · Automatisierte Erzeugung von Images ohne diese verteilen zu müssen
- Flexibilität bei der Konfiguration und bei den benutzten Software-Versionen
- · Einfache Syntax und damit einfach einsetzbar

Befehl: docker build -t <ziel_image_name> <Dockerfile>

Das Dockerfile definiert Aufbau und Inhalt des Image.



Dockerfile Kommandos

Element	Meaning
FROM <image-name></image-name>	Sets to base image (where the new image is derived from)
MAINTAINER <author></author>	Document author
RUN <command/>	Execute a shell command and commit the result as a new image layer (!)
ADD <src> <dest></dest></src>	Copy a file into the containers. <src> can also be an URL. If <src> refers to a TAR-file, then this file automatically gets un-tared.</src></src>
VOLUME <container-dir> <host-dir></host-dir></container-dir>	Mounts a host directory into the container.
ENV <key> <value></value></key>	Sets an environment variable. This environment variable can be overwritten at container start with the –e command line parameter of docker run .
ENTRYPOINT < command>	The process to be started at container startup
CMD <command/>	Parameters to the entrypoint process if no parameters are passed with docker run
WORKDIR <dir></dir>	Sets the working dir for all following commands
EXPOSE <port></port>	Informs Docker that a container listens on a specific port and this port should be exposed to other containers
USER <name></name>	Sets the user for all container commands

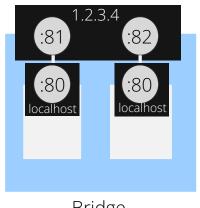
Typische Kommandos eines Docker Workflows

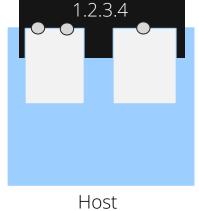
Command	Action
docker build -t <image/> .	Build Docker image from "Dockerfile" with given tag in current directory
docker images	Prints all local images
<pre>docker run -d -v <volume mounts=""> -p <host-port>:<container-port> <image/> <entrypoint process=""></entrypoint></container-port></host-port></volume></pre>	 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 optional entrypoint process)
<pre>docker run -ti <image/> /bin/sh</pre>	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</container>	Store container as local image
<pre>docker kill <container> docker rm <container></container></container></pre>	Terminate container (send SIGKILL to entrypoint process) Remove container
docker rmi -f <image/>	Remove local image

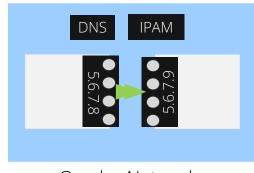
Hilfreiche Kommandos für Container Troubleshooting

Command	Action
docker inspect <container></container>	Shows container metadata (e.g. IP)
docker logs <container></container>	Prints container syslog
docker top <container></container>	Prints all running processes within a container (like ps –a within the container)
<pre>docker exec -ti <container> /bin/sh</container></pre>	Connect terminal to running container
docker stats <container></container>	Shows container runtime statistics (e.g. CPU usage, IO intensity,)
docker system prune	Removes all stopped containers, all unused images and all unused volumes
docker history <image/>	Show the Dockerfile commands for each image layer

Die Docker Networking Modes. • Docker erlaubt ein getrenntes Netzwerk zwischen docker containern aufzubauen.







Bridge

Overlay Network

docker network ls

docker network **inspect** bridge

docker network create --driver overlay multi-host-network

docker network connect multi-host-network container1

Bound port

Network interface

Guest

Host

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

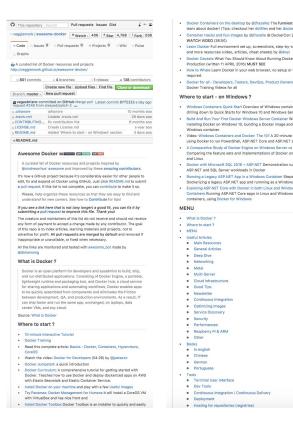
Reverse Proxy

Web Interface

Useful Imanes

Local Container Manager

Volume management and plugins



t way to erfiles)		
2015 MUST		
step tutorial		
er In		
installation		
s + Videos		
iners, erver 2016 Valkthrough d running a		
overview,		
Linux n Windows		
unning		
s for vs container sws s		

	0	Useful Images
	0	Dockerfile
	0	Docker Compose file
	0	Storing Images and Registries
	0	Monitoring
	0	Networking
	0	Logging
	0	Deployment and Infrastructure
	0	PaaS
	0	Remote Container Manager / Orchestration
	0	Security
	0	Service Discovery
	0	Metadata
	Slic	les
	Vid	eos
	0	Main Account
	0	Useful videos
	Inte	eractive Learning Environments
		eresting Twitter Accounts
		People
	Co	mmunities and Meetups
		LA ALL
S	ет	al Articles
a	in R	esources
		cker Weekly Huge resource
		cker Cheat Sheet by @wsargent MUST SEE
		cker Printable Refcard by @dimonomid
		nturyLink Labs
	Val	uable Docker Links Very complete
	Do	cker Ecosystem (Mind Map) MUST SEE
	Do	cker Ecosystem (PDF) MUST SEE find it on blog by Bryzgalov Peter.
	Blo	g of @frazelledazzell
	Blo	g of @jpetazzo
	Blo	g of @progrium
	Blo	g of @jwilder
	Blo	g of @crosbymichael
	Blo	g of @gliderlabs
	Blo	g of @sebgoa
	Blo	g of @codeship
	Dio	ital Ocean Community
		ntainer42
	Col	ntainer solutions
		ckerOne Docker Community (in Chinese) by @LiYingJie
	Pro	ject Web Dev : (Article series) How to create your own website based Docker
		oker vs. VMs? Combining Both for Cloud Portability Nirvana
		cker Containers on the desktop by @ifrazelle The funniest way to
	lea	rn about docker! (Tips: checkout her dotflies and her dockerfiles)) esome Linux Container more general about container than this repo,
	by	@Friz-zy.
1	nera	l Articles
		tting Started with Docker by @fideloper Servers For Hackers is uable resource. At some point, every programmer finds themselves eding to know their way around a server.
	nee	at is Docker and how do you monitor it?
	nee Wh	at is Docker and how do you monitor it? w to Use Docker on OS X: The Missing Guide

	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
	Snathanleclaire
	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
	Swarm v. Fleet v. Kubernetes v. Mesos Comparing different orchestration tools. (written OCT 2015)
	Swarm v. Fleet v. Kubernetes v. Mesos Comparing different orchestration tools, (written OCT 2015) The Shortlist of Docker Hostling There are so many specialized and optimized Docker hostling services available, it's high time for a review to
et	Swarm v. Fleet v. Kubernetes v. Mesos Comparing different orchestration tools, (written OCT 2015) The Shortlist of Docker Hostling There are so many specialized and optimized Docker hostling services available, it's high time for a review to see what's on offer (by Chris Ward).
et	Sowm v. Frest v. fubernetes v. Nelsos Comparing different orchestration tools, kurtisen OZ 1951619; There are so many specialized and postmised Docker hosting services available, if; it is high time for a review to see what so notifier (b) Christ Ward), species Articles time réplish introduçõe ao Docker e instateção no Ubuntu Que de uma integram eo que de um constanter Docker?
et	Sizemon Lifest V., Kalemietes v. Mesos Comparing different orchestration loss, letties Out 2019. This Shortist of Docker Hosting Them are so many seculiated and optimized Docker Hosting Them are so on many seculiated and optimized Docker Hosting Services available, it is high time for a review to upuse Articles Uses rigidal introdução ao Docker e Instalação no Ubunto Que de usa inagem e o que é um constairer Docker? Cindou los misegams e que é um constairer Docker?
et	Sowm v. Frest v. fubernetes v. Nelsos Comparing different orchestration tools, kurtisen OZ 1951619; There are so many specialized and postmised Docker hosting services available, if; it is high time for a review to see what so notifier (b) Christ Ward), species Articles time réplish introduçõe ao Docker e instateção no Ubuntu Que de uma integram eo que de um constanter Docker?
et	Sizemon Lifest V., Kalemietes v. Mesos Comparing different orchestration loss, letties Out 2019. This Shortist of Docker Hosting Them are so many seculiated and optimized Docker Hosting Them are so on many seculiated and optimized Docker Hosting Services available, it is high time for a review to upuse Articles Uses rigidal introdução ao Docker e Instalação no Ubunto Que de usa inagem e o que é um constairer Docker? Cindou los misegams e que é um constairer Docker?
et	Situation Lifest V., Indiametera v. Manos Comparing different orchestration consideration CD 2019. This Bibordini of Ducker Houstling There are an on many respectived and continued Ducker Houstling There are a so many respectived and continued Ducker Houstling services available, it is high time for a review to use enhants on offer (by Chris Ward). uguese Articles Unguese Articles Og quel d'uni minispam e o quale d'un condainer Docker? Cindrad uni minispam four guale d'un condainer Docker? Controlle uni minispam four plus de l'un condainer Docker? Controlle uni minispam four plus de l'un condainer Docker?
et	Sizemon Lifest V., Kalemietes v. Mesos Comparing different orchestration loss, Nettino CT 2019. This Shortist of Ducker Hosting Them are so many seculiated and optimized Ducker hosting Them are so many seculiated and optimized Ducker hosting services available, it is high time for a review to see what's on offer by Chris' Ward). upquese Articles unguese Articles Unguese and Comparing services and services of the Comparing services on the contract of Comparing services and services are policy Comparing services and services of body posts Comparing services and services of body posts (Comparing containers – Part 1 This is part one of a series of blog posts (Comparing containers – Part 1 This is part one of a series of blog posts (Comparing containers – Part 1 This is part one of a series of blog posts (Comparing containers – Part 1 This is part one of a series of blog posts (Comparing containers – Part 1 This is part one of a series of blog posts)
er	Summ. J. Red. V., Silammeter v., Mercha Comparing different orchestration locks, fereither CDT 2015. This shortest of Doctor Hosting Them are so many sescilarized and optimized Doctor hosting Them are so many sescilarized and optimized Doctor hosting average sold to the sold to be seen what so, or offer (by Chris' Ward). upwee Articles Uman facilia introducija o Doctor in satisfacjin on Ulturiti O and 4 mai Imagem e o que 4 um continiere Doctor? Doctor and in singen e o que 4 um continiere Doctor? Doctor and satisfacjin on Doctor in proposition Doctor and satisfacjin on Doctor in the satisfacjin on Ulturiti Doctor Doctor Company (1998). This is part one of a series of blog posts destalling how doctor creates containers. By disconstynicized
er	Sizenow J. Eleck V. Kildermeter v. Metros Comparing different orchestration foliation (Interface CD 2015): This Shorestia of Doctaer Housting There are so many secessized and optimized Doctaer House previous available, it is high time for a review to see what's on offer by Chris' Ward). Upper Articles: Uma rigidal introdução ao Doctaer e Instalação no Ubunto, O que é uma inagem e o que é um constainer Doctaer? Orcinado uma inagem e o que é um constainer Doctaer? Orcinado uma inagem e o que é um constainer Doctaer? Orcinado uma inagem e para é um constainer Doctaer? Orcinado uma inagem e para é un troute de a series el biog posts destingin por doctaer cartalers se, tyl Grosslymichael Data-only container madress working.
re	Journa D. Red X. Kalaminetes V. Meros Comparing different orchestration tools, feritation CD 2016. The size of souther produced and optimized Doods incharge services available, it is high time for a review to see what's on other by Chris Ward), queues Articles Uman ériplist introduções o Dooker a Instalação no Ubunho, que a uma insegem a que d'um container Dooker? Cristodo uma insegem Docker presentalação Comandos mais utilizados no Docker Do Docker Contrado uma insegem Docker personalizado Comandos mais utilizados no Docker Do Docker Contrado cuma contrador por Contrado cuma contrador por Contrado cuma contrador por Contrado cuma contrador por Contrado contrador contrador so. Se contrador contrador Contrador contrador Contrador
et	Journs D. Elec X. Submitted x. Metros Comparing different orchestration for the Comparing of the Section of Decision of Decis
et	Journey Lete V., Robernstein V., Messico Comparing different orchestration took (settles ACT 2018). The Bhoristic of Doctor Hosting Them are so many specialized and optimized Doctor hosting Them are as on many specialized and optimized Doctor hosting provides available, it is high time for a review to see what or on offer by Chris Wassil. Uses a shade on offer by Chris Wassil. Uses a shade on Doctor in Hosting Doctor of Doctor of Control uses an image of Doctor of Hosting Doctor of Control uses an image of Doctor of Hosting Doctor of Doctor of Christian Christian of Doctor of Doctor of Christian of Doctor of Christian of Doctor of Christian of Doctor of Doctor Doc
er	Journey - Elect - Kalaminetes - Mestros Comparing different orchestration forces (written CDT 2015). The Brothest of Doctor Hosting Them are so many specialized and optimized Doctor Hosting Them are as on many specialized and optimized Doctor Hosting Services available, it is high time for a review to see what is no rifer by Chris Ward. Uman signification on Doctor in Installação no Ulborato. Our ad umin imagem a quel su microstiture Doctor? Cristod uma imagem Doctor personalizado Commados mais utilizados no Doctor in Doctor Doctor? Cristod uma imagem Doctor personalizado Commados mais utilizados no Doctor Indiana de Comparis Compari
ert	Summ v. Elect v. Subminete v. Mercia Comparing different orchestration (soliton) (CTS 1951) This Shortist of Docker Housting Them are so many seculizated and optionized Docker Housting Them are no many seculizated and gottless and control of the CTS 1951. This Shortist of Docker Housting Them are not many seculizated and gottless of the CTS 1951. See what's on offer by Chris Ward, Uses enhants on offer by Chris Ward, Uses enhants on offer by Chris Ward, Uses enhants on Other by CTS 1951. On a stain minigrang to pay 4 un container Docker? Christic use minigrang Docker personalizated COntradio main stallization on Color and CTS 1951. Christic use stallization on Color and CTS 1951. Christic use stallization on Color and CTS 1951. Christic use on CTS 1951. Divide Container maintenance containers. By Grossbyreinchael Class - only container maintenance Color and CTS 1951. Lincip Docker Carelate containers. By Grossbyreinchael Class - Otto CTS 1951. Lincip Docker User Stallization of CTS 1951.
ert	Journa V. Bert V. Klammeter v. Messca Comparing different orchestration took (settles of C27 SI). The size of some special set of bodder some the Shortist of Doctar institute, it is a six of the size of the s
et	Journal De Hart K. Kalaminetes V. Meters Comparing different orchestration forces (written CD 270 Meters). The Brothest of Docker Hosting Them are so many specialized and optimized Docker hosting Them are a so many specialized and optimized Docker hosting Service available, it is high met for a review to see what's or offer by Chris Ward. Uman singles introducija on Docker in instalação no Ulumno Que a ulum iniquem a que d'um contalação no Ulumno Que a ulum iniquem a que d'um contalação Commando mais distilação no Docker Docker Commando mais distilação no Docker Docker Contalog contalaries. Part 1 This is part one el a service of biog posts detailing hostinalies. Part 1 This is part one el a service of biog posts detailing hostinalies. Part 1 This is part one el a service of biog posts detailing hostinalies. Part 1 This is part one el a service of biog posts detailing hostinalies. Part 1 This is part one el a service of biog posts detailing hostinalies. Part 1 This is part one el a service of biog posts detailing hostinalies. Part 1 This is part one el a service o biographical biodro docker constalar medieses working. Using Docker Machine with Wesse 0 10 (written 2 2 APR 2015) compost pooler overtay networking, by @nigelpoulton il divor to use Docker on Full Metall Cargotto A new essential OS for running the Docker Engine on bare matter of Dout. A Docker and mini-Past by @protogic
et	Journal De Mar L. Kalaminetes v. Mercis Comparing different orchestration (solic (incited DC 27 Miles). Comparing different orchestration for the Shortest of Docker Hosting. There are so many specialized and optimized Docker hostings There are so many specialized and optimized Docker hostings generate available, it is high time for a review to see what's an order by Oriz's Ward. Under a branching of the Comparing of the Compari

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 @iwilder · 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 Noiny Provy for Docker (written 9, II II, 2015). . Dealing with linked containers dependency in docker-compose by Docker Tine by @imervine. . Docker on Windows behind a firewall by @kaitoedter Pulling Git into a Docker image without leaving SSH keys behind by . 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 @losdirksen . 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 Production Meteor and Node Using Docker, Part I by @projectricochet · Resource Management in Docker by @marekgoldmann Newsletter Docker Team Tutum Shippable WebOps weekly . Docker and Phoenix: How to Make Your Continuous Integration More Jenkins 2.0 - Screencast Series by Virendra Bhalothia Pushing to ECR Using Jenkins Pipeline Plugin by @mikesir87 · Creating a Docker image from your code Ontimizing Docker Images . How to Optimize Your Dockerfile by @tutumcloud . Building Docker Images for Static Go Binaries by @kelsevhightower Squashing Docker Images by @iwilder . Dockerfile Golf (or optimizing the Docker build process) DockerStim shrinks fat Docker images creating the smallest possible. . SkinnyWhale Skinnywhale helps you make smaller (as in megabytes) Docker containers. . MicroBadger - Analyze the contents of images and add metadata labels . @progrium Service Discovery articles series: Consul Service Discovery with Docker Understanding Modern Service Discovery with Docker Automatic Docker Service Announcement with Registrator