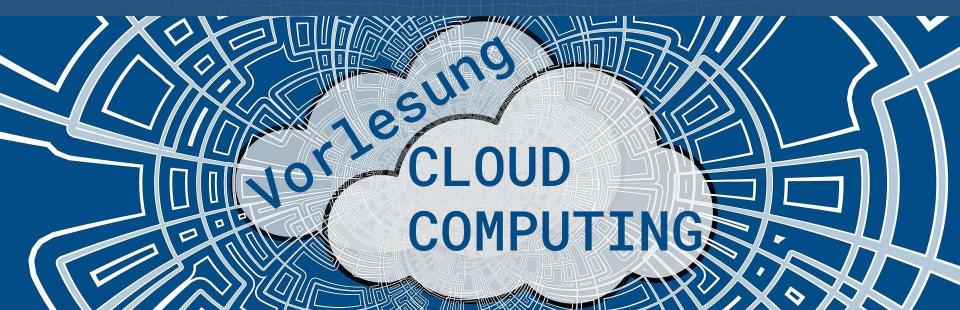


Kapitel 3: Virtualisierung



Anforderungen an Cloud Computing

- 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 vermeiden. Security
 - Entkopplung von der Hardware für mehr Flexibilität im Betrieb und Robustheit bei Ausfällen
 - · Ressourcen sollen flexibel vergeben werden. Steuerung mittels Software defined ressources
- · Lösung für diese Anforderungen ist Virtualisierung 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



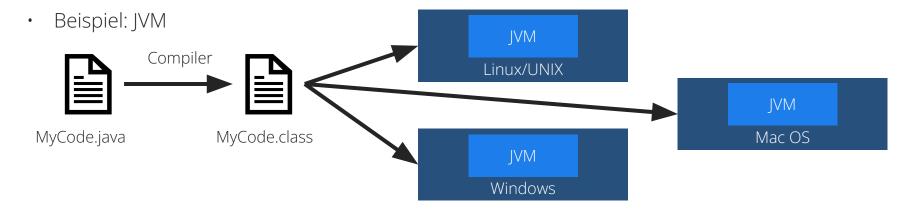
Zur Vollständigkeit: Was ist Anwendungs-Virtualisierung?

 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.
 - $x86 \text{ Host} \rightarrow x86 \text{ Guest}$

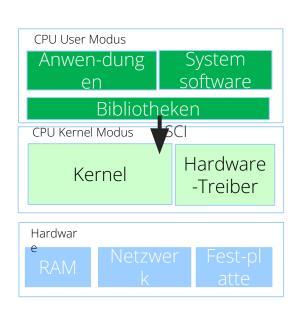
Klassischer Aufbau eines Betriebssystems mit Unterstützung der Rechnerarchitektur

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

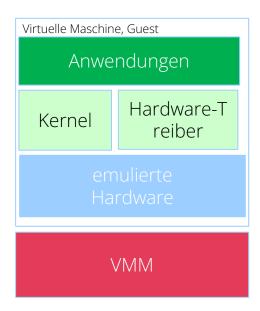
• Eine lauffähige / laufende virtuelle Maschine

VMM (Virtual Machine Monitor, auch Hypervisor genannt)

 Die Steuerungssoftware zur Verwaltung der Guests und der Host-Ressourcen

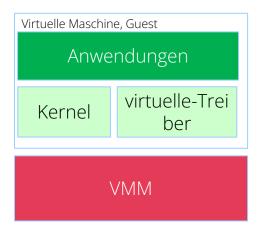
Voll-Virtualisierung

- 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%



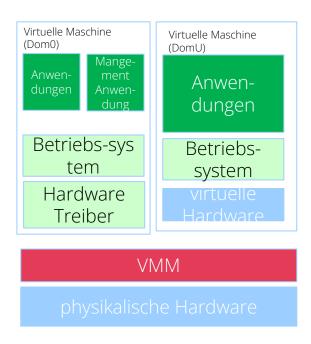
Para-Virtualisierung

- 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%



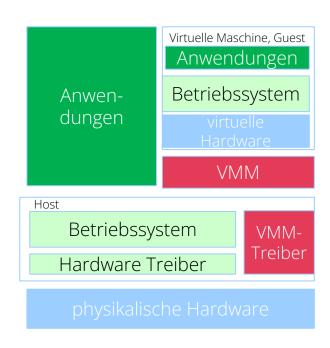
Typ 1: Bare-Metal Virtualisierung

- 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 Balooning Hauptspeicher dynamisch vergrößern
 - · 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



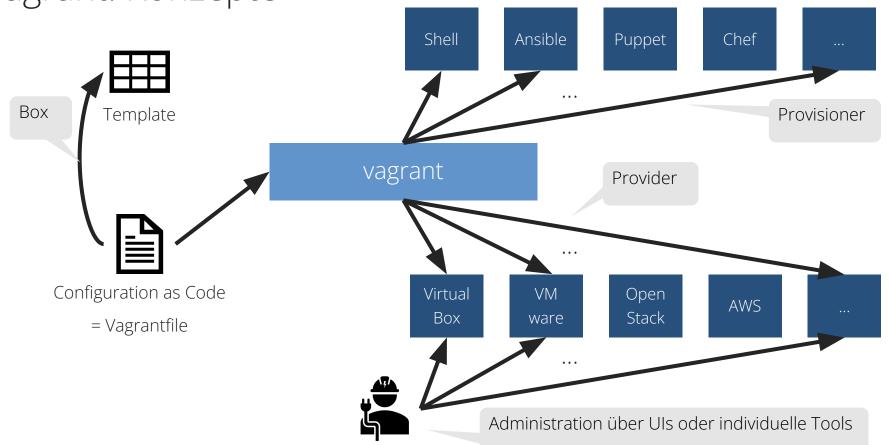




Open Source Typ 2 Virtualisierungs-Software (Voll-Virtualisierung) für Windows, Linux, MacOS und Solaris.

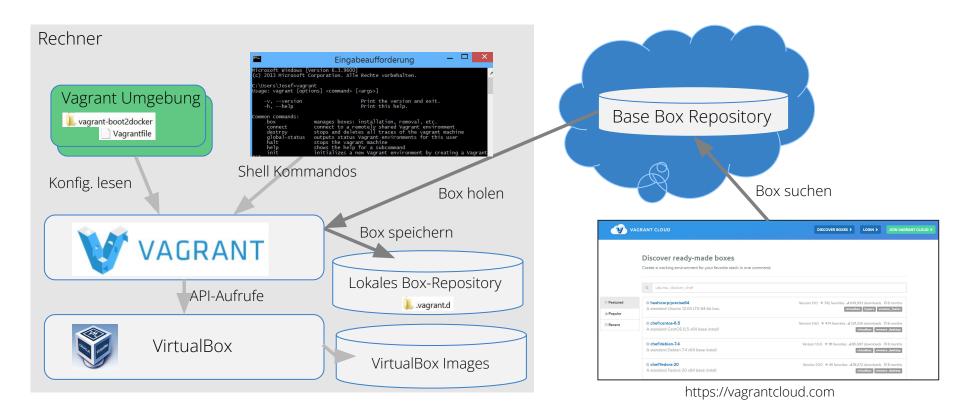
Automationssoftware für virtuelle Umgebungen auf einem Rechner. Virtuelle Maschinen per Kommandozeile erstellen und steuern.

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.
- 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.
- <u>vagrant < command > -help</u> 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.
- Kompatibiltät 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

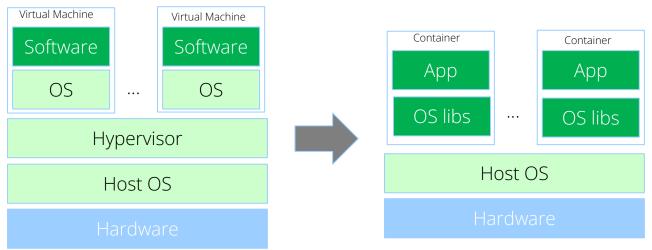
Linux Kernel Namespaces (Jahr 2002)

- 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

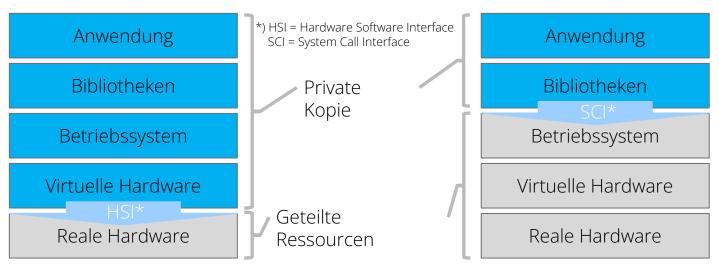


- Leichtgewichtiger Virtualisierungsansatz: Es gibt keinen Hypervisor. Jede App läuft direkt als Prozess im Host-Betriebssystem. Dieser ist jedoch maximal durch entsprechende OS-Mechanismen isoliert (z.B. Linux LXC).
 - Isolation des Prozesses durch Kernel Namespaces (bzgl. CPU, RAM und Disk I/O) und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

Kontainerisierung 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 (2013)

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.

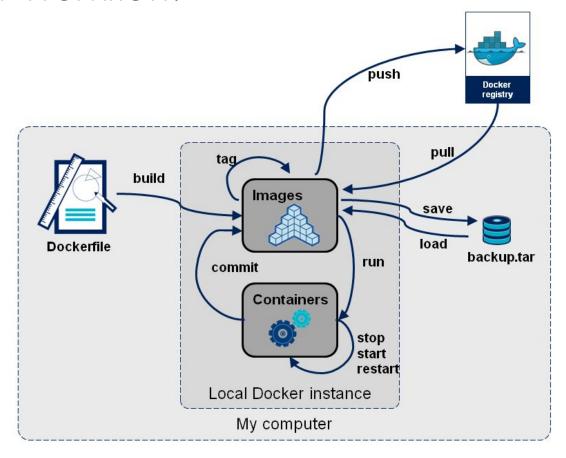
Docker

- 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.

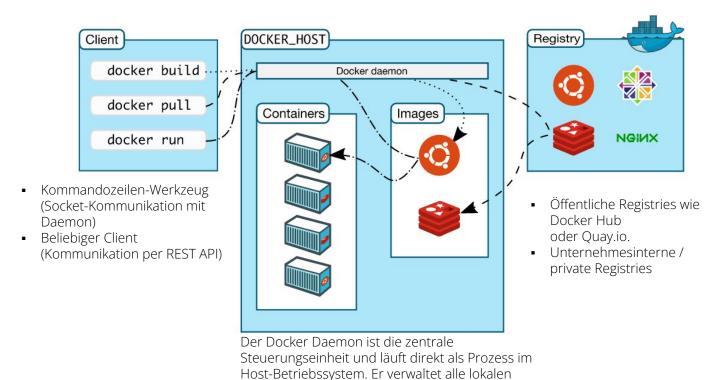


https://www.openhub.net/p/docker

Der Docker Workflow.

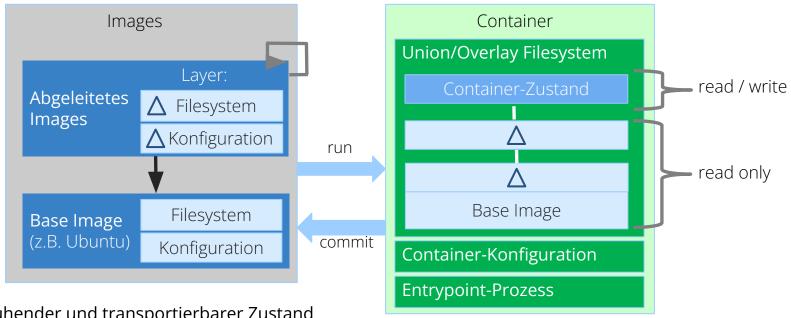


Die Docker Architektur.



Container und Images auf dem Host.

Im Zentrum von Docker stehen Images und Container.



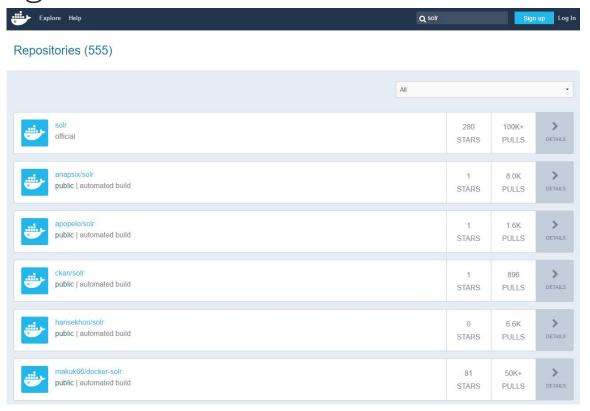
Ruhender und transportierbarer Zustand

- 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.



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

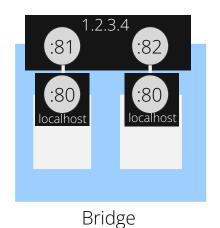
Typische Kommandos eines Docker Workflows

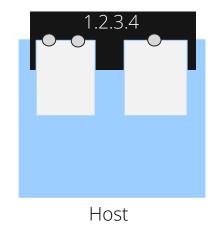
Command	Action
docker build -t <image/> .	Build Docker image from "Dockerfile" with given tag in current directory
docker images	Prints all local images
<pre>docker run -d -v <volume mounts=""> -p <host-port>:<container-port> <image/> <entrypoint process=""></entrypoint></container-port></host-port></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 $-a = only running containers)$
<pre>docker commit <container> qaware/foo</container></pre>	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

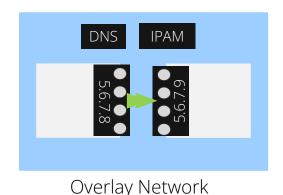
Hilfreiche Kommandos für Container Troubleshooting

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.





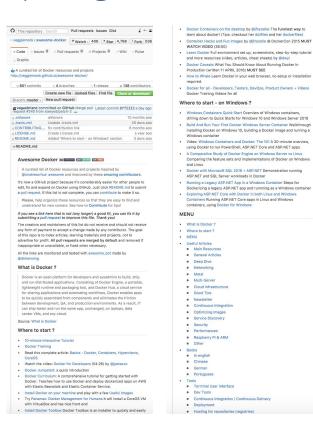


docker network ls docker network **inspect** bridge docker network create --driver overlay multi-host-network docker network connect multi-host-network container1

Bound port Network interface Guest Host



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



	Reverse Proxy
	Web Interface
	Local Container Manager
	Volume management and plugins
	Useful Images
	Dockerfile
	Docker Compose file
	Storing Images and Registries
	Monitoring
	Networking
	• Logging
	Deployment and Infrastructure
	• PaaS
	Remote Container Manager / Orchestration
	Security
	Service Discovery
	Metadata
	Slides
	Videos
	Main Account
	Useful videos
	Interactive Learning Environments
	Interesting Twitter Accounts
	• People
	Communities and Meetups
Jse	eful Articles
Mair	n Resources
	Docker Weekly Huge resource
	Docker Cheat Sheet by @wsargent MUST SEE
	Docker Printable Refcard by @dimonomid
	CenturyLink Labs
	CenturyLink Labs Valuable Docker Links Very complete
	Valuable Docker Links Very complete Docker Ecosystem (Mind Map) MUST SEE
	Valuable Docker Links Very complete Docker Ecosystem (Mind Map) MUST SEE
	Valuable Docker Links Very complete Docker Ecosystem (Mind Map) MUST SEE Docker Ecosystem (PDF) MUST SEE find it on blog by Bryzgalov Peter.
	Valuable Docker Links Very complete Oocker Ecosystem (Mind Map) MUST SEE Docker Ecosystem (PDF) MUST SEE find it on blog by Bryzgalov Peter. Blog of @frazelledszzell Blog of @ipretazzo
	Valuable Docker Links Very complete Docker Ecosystem (Mind Map) MUST SEE Docker Ecosystem (MCP) MUST SEE find it on blog by Bryzgalov Peter. Blog of @prazelledszzell Blog of @pretazzell Blog of @pretazzell
	Valuable Docker Links Very complete Oocker Ecosystem (Mind Map) MUST SEE Docker Ecosystem (PDF) MUST SEE find it on blog by Bryzgalov Peter. Blog of @frazefiledszzell Blog of @ipetazzo
	Valuable Docker Links Very complete Docker Ecosystem (full Mada) MLJT SEE Docker Ecosystem (PDF) MLJST SEE find it on bing by Bryzgalov Peter Blog of Giffexelledazzell
	Valuable Decleric Hink Very compilete Obsolete Ecosystem Mich May MMIT SEE Docket Ecosystem MPD/ MUST SEE find it on blog by Bryzgalov Peter. Blog of Giffeszeindazzei Blog of Giffeszeindazzei Blog of Giffeszeindazzei Blog of Blogoriyam Blog of Blogoriyam Blog of Giffeszeindazzei Blog of Giffeszeindazzei Blog of Giffeszeindazeind
	Valuabie Decisier Unix Very compilete Doctoner Ecosystem MUN Mod pM MUT SEE Doctoner Ecosystem MUN Mod pM MUT SEE Doctoner Ecosystem MUN MOD SEE Doctoner Georgestem GUN MUN SEE Doctoner Georgestem GUN MUN SEE Doctoner GUN GUN GUN GUN GUN GUN GUN DOCTONER
	Valuabia Decisier Links Very compilete Docknet Ecosystam (MW May MMIT SEE Docknet Ecosystam (MW May MMIT SEE Docknet Ecosystam (MW May MMIT SEE Bloop of Giffentation
	Valuabie Decisier Unix Very compilete Doctoner Ecosystem MUN Mod pM MUT SEE Doctoner Ecosystem MUN Mod pM MUT SEE Doctoner Ecosystem MUN MOD SEE Doctoner Georgestem GUN MUN SEE Doctoner Georgestem GUN MUN SEE Doctoner GUN GUN GUN GUN GUN GUN GUN DOCTONER
	Valuabie Declaric Unix Very compilete Doctoner Ecosystem MUN Mody MUNT SEE find it on blog by Bryzgałov Peter. Doctoner Conystem MUPT MUST SEE find it on blog by Bryzgałov Peter. Digo of Gillezelede
	Valuable Declaric Units Very complete Doctore Ecosystem (MW Map) MUST SEE find it on blog by Bryzgalov Peter. Doctore Ecosystem (MPD) MUST SEE find it on blog by Bryzgalov Peter. Digo of differentiated Digo
	Valuable Decisir Links Very compilete Doctoner Ecosystem (DVM May) MUST SEE Doctoner Ecosystem (DVM May) MUST SEE Doctoner Ecosystem (DVM May) SEE Doctoner Ecosystem (DVM May) Doctoner Ecosystem (DVM May) Doctoner Docto
	Valuable Decisir Links Very compilete Doctoner Ecosystem (MW May) MUST SEE find it on blog by Bryggalov Peter. Doctoner Ecosystem (PDP) MUST SEE find it on blog by Bryggalov Peter. Digs of differentiated (Bulgo of differentiated (Bulgo of differentiated) Bulgo of disposition Documentor Doctor Community (or Chinesel by BUTring, Ille Project Vive Der : (Intrice series) How to create your own website base on Oscider
	Valuable Decisite Units Very compilete Doctore Ecosystem MOVM May MUST SEE find it on blog by Brygaplov Peter. Doctore Ecosystem MOVP MUST SEE find it on blog by Brygaplov Peter. Doctor Consultation of Cons
	Valuable Decisir Unix Very compiles Doctore Ecosystem MWM Map MMITS SEE find it on blog by Brygajev Peter. Blooker Ecosystem WDF MMITS SEE find it on blog by Brygajev Peter. Blog of Giffezelledesidesidesidesidesidesidesidesidesides
	Valuable Decisir Unix Very compiles Doctore Ecosystem MWM Map MMITS SEE find it on blog by Brygajev Peter. Blooker Ecosystem WDF MMITS SEE find it on blog by Brygajev Peter. Blog of Giffezelledesidesidesidesidesidesidesidesidesides
	Valuable Decisir Unix Very compilete Doctore Ecosystem (MW Map) MUT SEE find it on blog by Bryzgałov Peter. Doctore Ecosystem (DPI) MUST SEE find it on blog by Bryzgałov Peter. Digo of diffezelledezing Digo of diffezelled
Sen	Valuable Decisit Links Very compilete Doctober Ecosystem MOVM May MUST SEE find it on blog by Brygaplov Peter. Doctober Ecosystem MOVD MUST SEE find it on blog by Brygaplov Peter. Doctober Coopystem MOVD MUST SEE find it on blog by Brygaplov Peter. Doctober Coopystem MOVD MUST SEE find it on blog by Brygaplov Peter. Doctober Coopystem MUST SEE find it on blog by Brygaplov Peter. Doctober Coopystem MUST SEE find it on blog of Berdaplos Doctober Coopystem MUST SEE find it on the Coopystem MUST Coopystem About Coopystem August MUST SEE find It on the Coopystem MUST SEE find It on
3 Sen	Valuable Decisit Links Very compiles Doctore Ecosystem MOVM May MMT SEE Doctore Ecosystem MOVM May MMT SEE Doctore Ecosystem MOVM MAY SEE find it on blog by Blyzgalov Peter. Doctor of Brezeledesian Doc of Blezeledesian Doctor of Blezeledesian
Sen	Valuable Decisit Links Very compilete Doctorer Ecosystem (MW Map) MUST SEE Doctorer Ecosystem (MW Map) MUST SEE Doctorer Ecosystem (MW Map) MUST SEE Doctorer Ecosystem (MW MAP) Decisit See Doctor Se
Sen	Valuable Desire Link Very complete Doctore Ecosystem (DFI) MMST SEE find it on blog by Bryggloov Peter. Doctore Ecosystem (DFI) MMST SEE find it on blog by Bryggloov Peter. Doctor of Ecosystem (DFI) MMST SEE find it on blog by Bryggloov Peter. Doctor of Ecosystem (DFI) MMST SEE find it on blog by Bryggloov Peter. Doctor of Ecosystem (DFI) MMST SEE find it on blog by Bryggloov Peter. Doctor of Ecosystem (DFI) MMST SEE find it on blog by Bryggloov Peter. Doctor of Ecosystem (DFI) MMST SEE find it on the SEE find it on the SEE find it of Ecosystem (DFI) MMST SEE find it on the SEE find it of Ecosystem (DFI) MMST SEE find it on the SEE find it of Ecosystem (DFI) MMST SEE find it on the SEE find it of Ecosystem (DFI) MMST SEE find it on the SEE find it of Ecosystem (DFI) MMST SEE find it on the SEE find i

	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).
	tuquese Articles
ol	togacoo ni tiolo
ol	Uma rápida introdução ao Docker e instalação no Ubuntu
ol	Uma rápida introdução ao Docker e instalação no Ubuntu O que é uma imagem e o que é um container Docker?
ol	Uma rápida introdução ao Docker e instalação no Ubuntu O que á uma imagem e o que á um container Docker? Criando uma imagem Docker personalizada
ol	Uma rápida introdução ao Docker e instalação no Ubuntu O que é uma imagem e o que é um container Docker?
	Uma rápida introdução ao Docker e instalação no Ubuntu O que á uma imagem e o que á um container Docker? Criando uma imagem Docker personalizada
	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
•	Uma rifolda introdução ao Docker e instalação no Ubuntu O que é uma imagem e o que é um container Docker? Cindio uma imagem booker personalizada Comandos mais utilizados no Docker po Dove Creating containers - Part 1 This is part one of a series of biog posts detailing nove docker creates containers. By directolyminimal Date and y containers and disease.
•	Una spida introdução ao Docker e hestaleção no Ubontu O gos de misegam no gos de un crestalent Docker? Comandos mais utilizados no Docker Oporandos no Doc
•	Uma rifolda introdução ao Docker e instalação no Ubuntu O que é uma imagem e o que é um container Docker? Cindado uma imagem bocker personalizada Comandos mais utilizados no Docker pe polve Constituiros — Burs 1. Tiba is part des el a series el biog posts detailing how docker creates containers. By (Boresbymichael Data-only, contrainer madiessa teorising Using Docker Machine with Viesee 0.10 (written 22 APR 2015)
•	Una spida introdução ao Docker e hestaleção no Ubontu O gos de misegam no gos de un crestalent Docker? Comandos mais utilizados no Docker Oporandos no Doc
•	Uma rádida introdução ao Docker e instalação no Ulporitu O que duma integram e o que du mocitalism Docker? Contrado uma imagem ao Que dum contalism Docker? Contrado que imagem docker permonástica. Contrados más utilizados no Docker po (Ne Creating container — Part 1 This is part one of a series of biog posts detailing how docker creates containers. By @crostlymichael Data-ody container macloses. Exercising Using Docker Mechine with Weleve 0.10 (written 22 APR 2016) Holing Docker Mechine with Weleve 0.10 (written 22 APR 2016)
e	Uma rifolda introdução ao Docker e instalação no Ubuntu O que é uma imagem e o que é um container Docker? Cindado uma imagem bocker personalizada Comandos mais utilizados no Docker per polive Constituers — Rer 1. This is part one of a series of blog posts detailing how docker creates containers. By (Borosbymichael Data-only contrainer madiessa teorxing Using Docker Machine with Vicene O.10 (written 22 APR 2015) Hour to Route Tariffic through a Tor Docker container by (gifraszele writtent 20 J.N. 2015)
e	Uma rádida introdução ao Docker e instalação no Ulporitu O que é uma insegem e o que é um container Docker? Crinado uma insegem so que é um container Docker? Crinado uma insegem soborter personativado Comandos mais utilizados no Docker po Dive Creating containers — Part 1 This is part one of a series of biog posts detailing how docker creates containers. By @creatyminhadil Data-only container madiosas Westernadosas Using Docker describe with Visione 0.10 (eritisse 22.AFS 2015) How to Bloats Traffic trougle 3 for Docker container by @fitzazele (evitisen 0.UM 2015) Demystifing Docker overlay networking. By @ingelpoution tol.
e	Uma rifolda introdução ao Docker e instalação no Ubontu O que é uma imagem e o que é um container Docker? Cindado uma imagem bocker personativado Comandos mais utilizados no Docker por Dive Cindado que imagem por porte por porte por porte por porte porte por porte por porte
e	Uma rigidal introdução ao Docker e instalação no Ubontu O que dum imagem e o que du montaliste Docker? Contrados mais introdução no Docker (Creating continers - Part 1 This is part one of a series of biog posts desting nove obocer creates containers, by discretisymichael Date-only container andiosas Norvicing Using Docker Machine with Viewe 0.10 (written 22 APR 2015) How to Booker Traffic through a Tic Docker container by gifferaselle (written 20 UN 2015) Demysión go bodos oversiny networking, By @injespoutton tal How to use Docker on Full Matal Cargodo A bare essentia OS for running the Docker Engine on bare metal or Could.
e	Uma rifolda introdução ao Docker e instalação no Ubontu O que é uma imagem e o que é um container Docker? Cindado uma imagem bocker personativado Comandos mais utilizados no Docker por Dive Cindado que imagem por porte por porte por porte por porte porte por porte por porte
e	Unia rápida introdução ao Docker e instalação no Ubontu O que dum inagum e o que dum container Docker? Comandos mais inagum e o que dum container Docker? Comandos mais utilizados no Docker pp Dive Constitução positioner - Part 1 This is part one of a series of blog posts decidang nove docker creates containers. By @crostlyminhale Dolke - dity consistem matiesse vocating Using Docker Mochine with Weeve 0.10 (written 22 APR 2015) Hoste in Bouter Tariffe through a Tico Docker container by gifyrassele (written 20 July 2016) Demystring Docker on Full Mattel Cargodo A bare essentia OS for running the Docker Engine on bare metal or Colour. Itili - Barver A Docker bassed mini-Pasal by @prologic
e	Uma rádida introdução ao Docker e instalação no Ulporitu O que é uma integem e o que é um contailer Docker? Cincado uma insegum e o que é um contailer Docker? Cincado uma insegum docker personalisado. Comandos más utilizados no Docker Do Dive Creating containers – Part 1 This is part one of a series of blog posts detailing how docker creates containers. By @crostlymitchael Data-only container madiosas Using Docker ocetainer madiosas Using Docker Machine with Wiseen 0.10 portina 22 APR 2015) Home of Booker Traitic through a Tor Docker container by gylfrasele (writtent 20 UNA 2015) Demystifring Docker overlay networking. By @inglepoutton tal How to use Docker on Full Matela Cappoló A bare essertial 05 for numining the Docker Engine on bare mittel of Coxes.

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 Tins 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 CenturyLink Labs Tutum Shinnahla 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

E 2