docker

Carsten Grohmann

25 March 2015

GLOBALFOUNDRIES

Agenda

1. Introduction
2. Live presentation
3. Discussion

Agenda

1. Introduction
2. Live presentation
3. Discussion

Docker is an application specific environment to automate and accelerate release cycles.

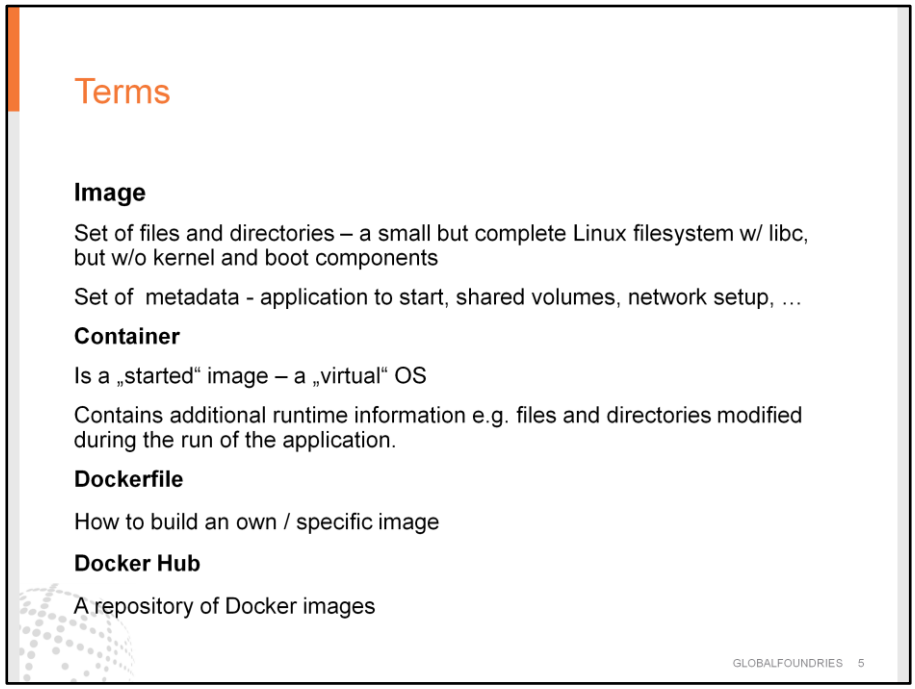
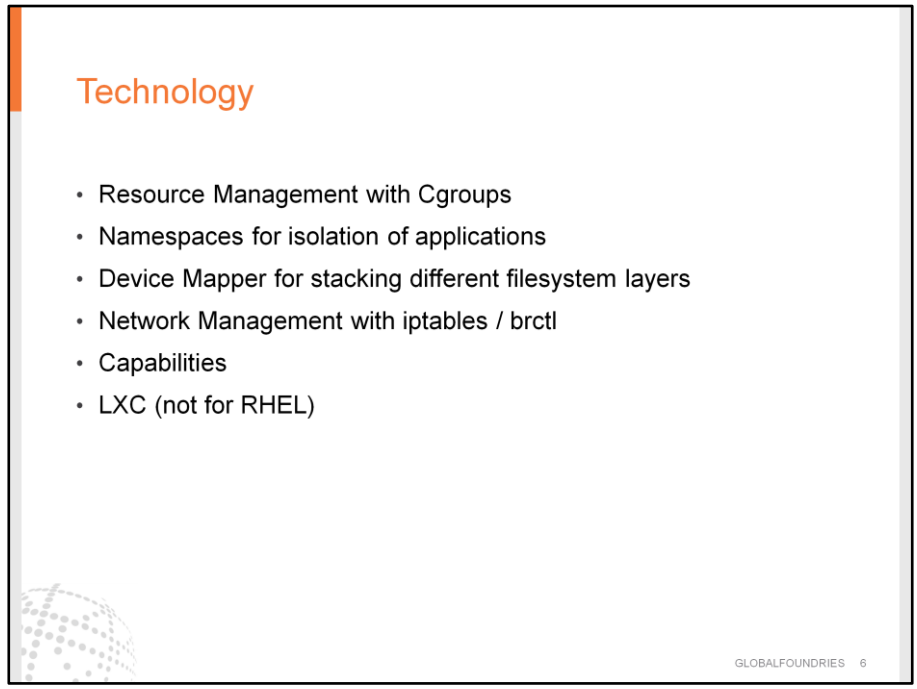


Image:

- Mehrere Ebenen
- Nicht veränderbar -> neues abgeleitetes Image anlegen.
- Änderungen zur Laufzeit eines Containers werden im Container gespeichert.

Container:

- System Container
  - mit init, inetd, syslogd, cron, …
- Application Container
  - nur die Anwendung
  - Keine „Verschwendung“ von Ressourcen wie RAM, CPU
  - Keine eigene IP-Adresse

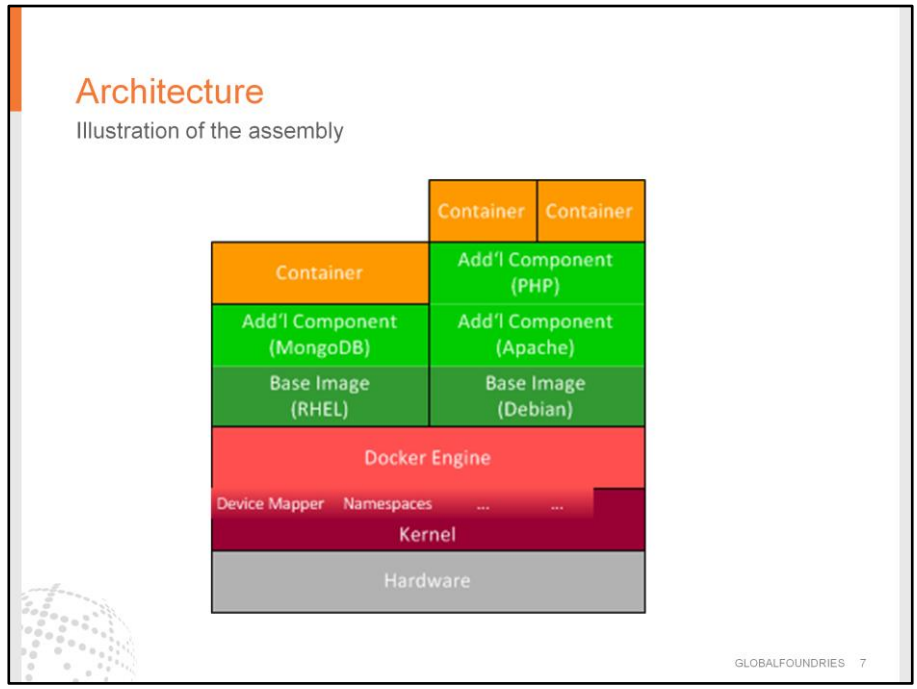


Cgroups – Control Groups:
- Verwaltung der Ressourcen von Prozessgruppen

Namespaces:
- Process namespace
- Network namespace
- IPC namespace (Semaphore, Shared Memory, Message Queues)
- Mount namespace
- UTS namespace (Unix Time Sharing) – Host- und Domainname)

Network Management:
- Jeder Container bekommt sein eigenes Netz aus RFC 1918 (10/8, 172.16/12 oder 192.168/16)
- Kommunikation erfolgt immer über Ethernet Bridges
- Ausgehende Verbindungen sind immer erlaubt
- Eingehende Verbindungen müssen dagegen explizit konfiguriert werden
- Abbilden von internen offenen Ports auf zufälligen Host-Ports
- Links zwischen zwei Containern
- Netzwerkstacks zwischen Container teilen
- Oder so ziemlich alles was man mit Firewall, Bridges und Name Spaces konfigurieren kann

LXC - zweideuting:
- Linux Container
- Tool „lxc“ zum Verwalten von Containern

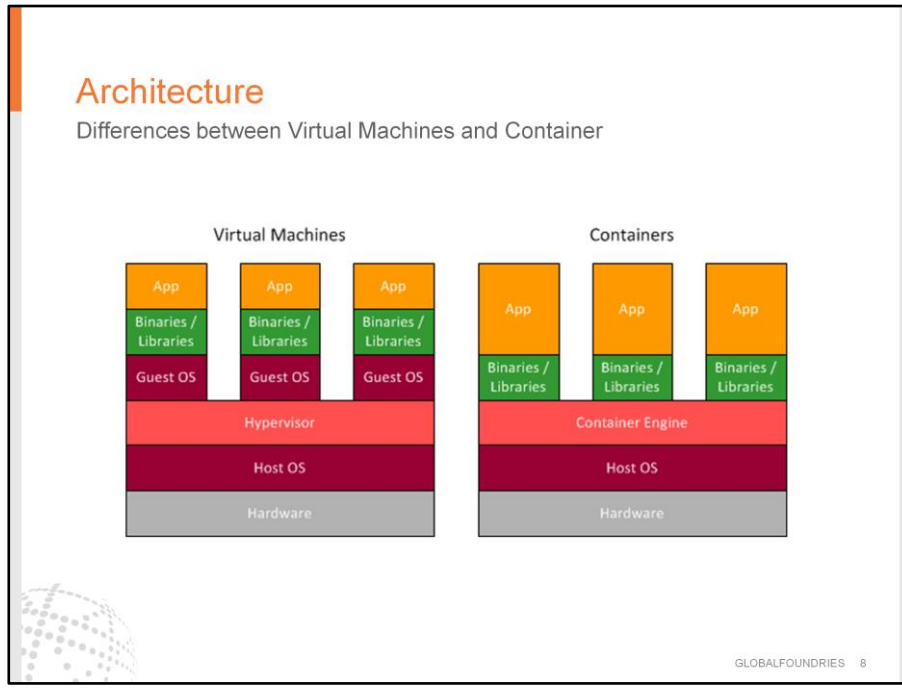


Schematischer Aufbau eines Docker Hosts:

- Hardware
- Kernel und dessen Komponenten
- Container Engine – Steuerung Komponenten
- Docker Image (dicke schwarze Linie), welches sich aus verschiedenen Schichten zusammensetzt
- Pro Schritt beim Erstellen eines Images entsteht eine Schicht
- Container, laufende Instanz eines Image, mit allen Änderungen, die zur Laufzeit am Image vorgenommen werden (z.B. Logfiles, …)

Farben:

- grau – Hardware
- Rottöne – Host OS
- Dunkelgrün – Basis Image
- Hellgrüne Töne – zusätzliche Ebenen durch Änderungen des Images
- Orange – laufenden Container



Zwei verschiedene Prinzipien

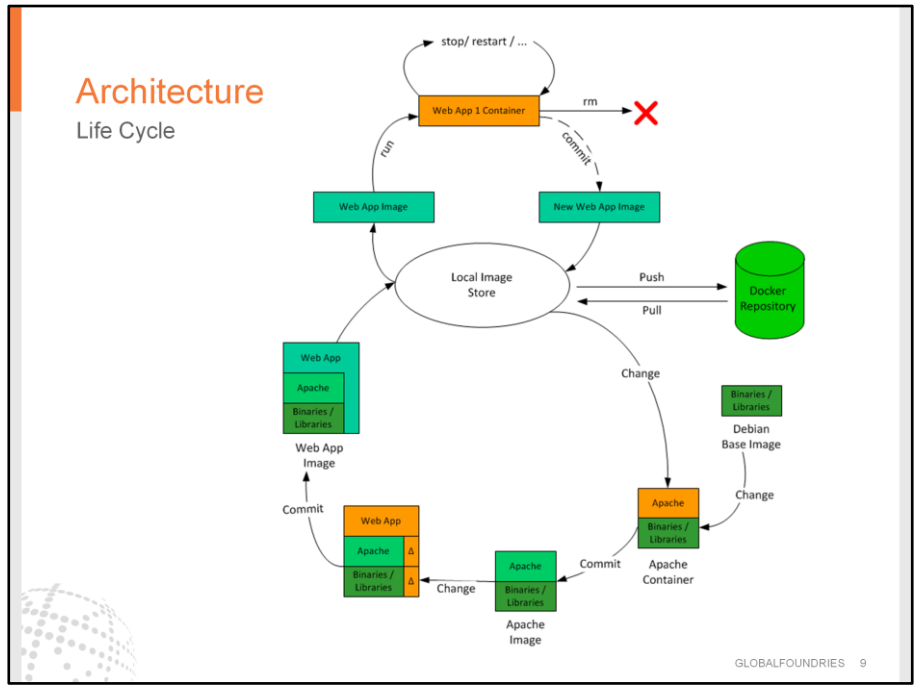
1. Aufteilung von großer Hardware in viele kleine Stücke
2. Separierung durch Fähigkeiten des Kernel (z.B. Namensräume, stacked filesystems)

Bei VMs

- Separierung durch die Simulation eigener Hardware
- Mehraufwand
    - Zerteilen der Hardwareressourcen und deren Zuweisung an einzelne VMs durch den Hypervisor
    - Zusätzliche HW-Ressourcen wie z.B. CPU und RAM für Hypervisior und Guest OS

Bei Containern

- Separierung durch Fähigkeiten des Kernel (z.B. Namensräume, Stacked Filesystems)
- Virtualisierung auf Prozessebene und nicht Hardwareebene
    - ein Kernel-Feature
    - keine extra Ressourcen notwendig
- Bessere Nutzung vorhandener Ressourcen möglich
- Nur wenig Mehraufwand durch Device Mapper und das Netzwerksetup (Firewall / Bridging)
- durch Namespaces laufen die Anwendungen mit nativer Performance



Farben:

- grau – Hardware
- Rottöne – Host OS
- Dunkelgrün – Basis Image
- Hellgrüne Töne – zusätzliche Ebenen durch Änderungen des Images
- Orange – laufenden Container

Agenda

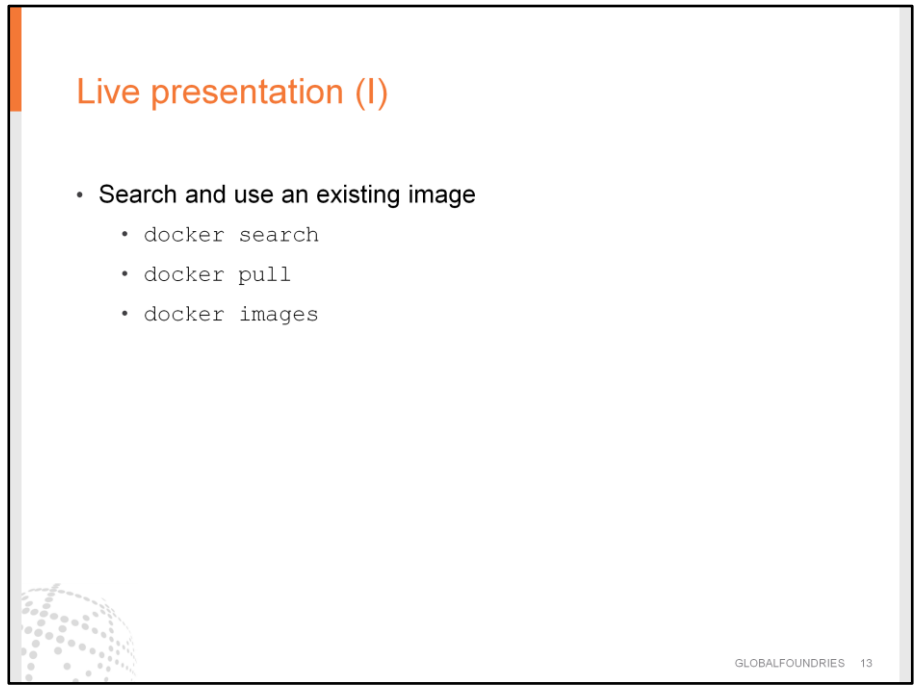
1. Introduction
2. Live presentation
3. Discussion

Installation on RHEL7

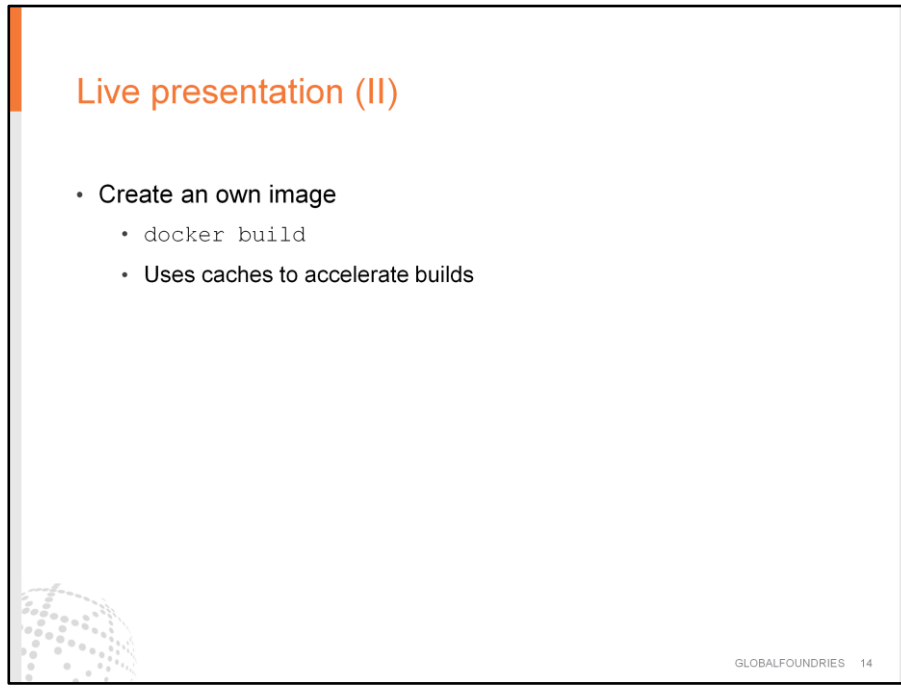
```
# yum install docker
# systemctl enable docker.service
# systemctl start docker.service
```

Installation on RHEL7

```
# docker version
Client version: 1.4.1-dev
Client API version: 1.17
Go version (client): go1.3.1
Git commit (client): d26b358/1.4.1
OS/Arch (client): linux/amd64
Server version: 1.4.1-dev
Server API version: 1.17
Go version (server): go1.3.1
Git commit (server): d26b358/1.4.1
```




```
# docker search centos
NAME                                DESCRIPTION                                     STARS
OFFICIAL   AUTOMATED
centos                              The official build of CentOS.                   751       [OK]
ansible/centos7-ansible             Ansible on Centos7                              26
[OK]

# docker pull centos:7
Pulling repository centos
8efe422e6104: Download complete
511136ea3c5a: Download complete
5b12ef8fd570: Download complete
Status: Downloaded newer image for centos:7

# docker images
REPOSITORY                      TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
centos                          latest              8efe422e6104        2 weeks ago         224 MB
centos                          7                   8efe422e6104        2 weeks ago         224 MB
centos                          centos7             8efe422e6104        2 weeks ago         224
MB
```



cat Dockerfile

docker build --tag centos:20150325 - < Dockerfile
Sending build context to Docker daemon 2.048 kB
Sending build context to Docker daemon
Step 0 : FROM centos:7
 ---> 8efe422e6104
[...]
Step 6 : RUN sed -i -e 's/UTC/LOCAL/' /etc/adjtime
 ---> Running in 5d3373e5afb8
 ---> 92bf5c101bb3
Removing intermediate container 5d3373e5afb8
Successfully built 92bf5c101bb3

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
centos	20150325	92bf5c101bb3	2 minutes ago	306.2 MB

Optional:
vi Dockerfile
docker build --tag centos:20150325_2 - < Dockerfile

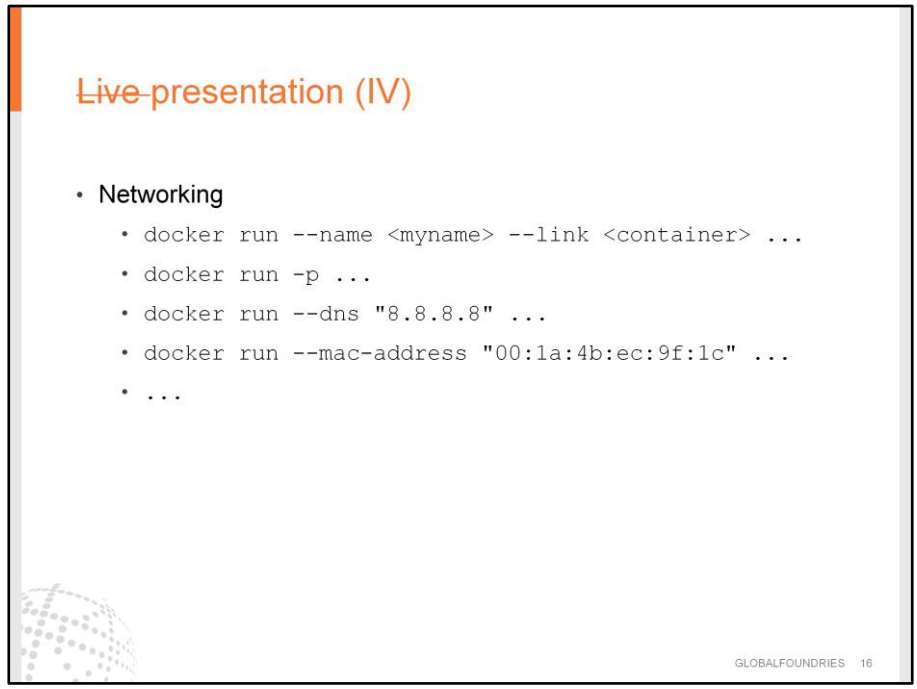
Build Cache:
Docker speichert die einzelnen Images, die durch die Abarbeitung der Schritte eines Dockerfiles entstehen, zwischen. Dadurch beschleunigen sich wiederholende Build-Vorgänge.
Bei Änderungen am übergeordneten Image werden alle nachfolgenden Images neu erstellt.

Live presentation (III)

- Run a container (Escape sequence: Ctrl-p Ctrl-q)
  - docker run
  - docker attach
  - docker stop

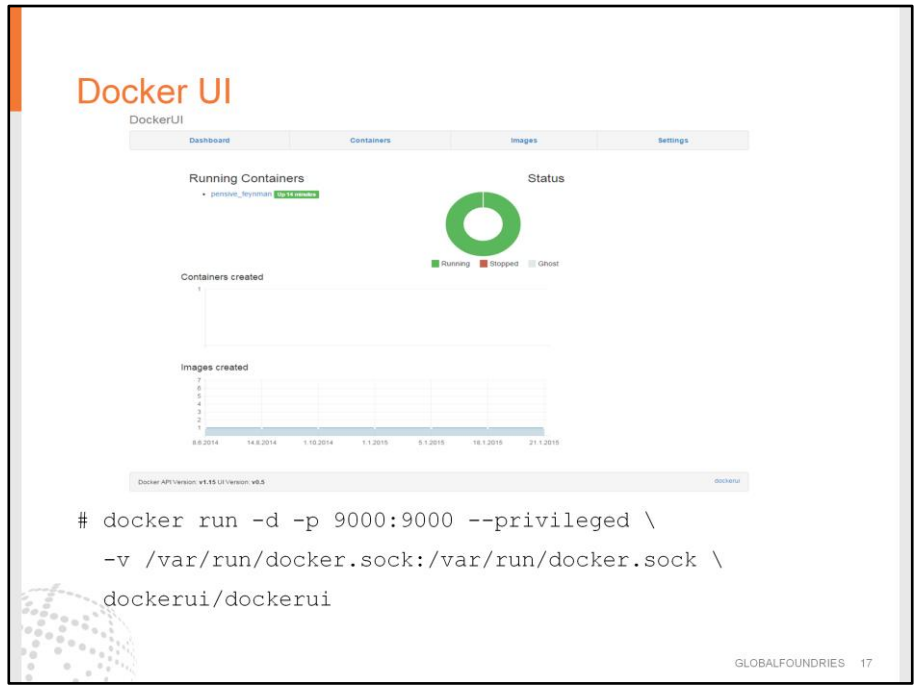
GLOBALFOUNDRIES 15

```
# docker run -i -t centos:20150325 /bin/bash
[root@4aff0dcccb40 /]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  1 18:35 ?        00:00:00 /bin/bash
root        17     1  0 18:35 ?        00:00:00 ps -ef
[root@4aff0dcccb40 /]# hostname
4aff0dcccb40
<abgetrennt mit Ctrl-p Ctrl-q>

# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
4aff0dcccb40        centos:20150325     "/bin/bash"         About a minute
ago   Up About a minute                       sad_darwin

# docker attach sad_darwin
[root@4aff0dcccb40 /]# uptime
 18:37:47 up 1 day,  2:44,  0 users,  load average: 0.00, 0.12, 0.26
[root@4aff0dcccb40 /]# exit
exit

Beendete Container existieren weiterhin:
# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES
4aff0dcccb40        centos:20150120     "/bin/bash"         3 minutes ago
Exited (0) 52 seconds ago                       sad_darwin

Und können neu gestartet werden:
# docker attach sad_darwin
2015/01/20 17:39:13 You cannot attach to a stopped container, start it
first

# docker restart sad_darwin
sad_darwin
# docker attach sad_darwin

[root@4aff0dcccb40 /]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 18:39 ?        00:00:00 /bin/bash
root        17     1  0 18:39 ?        00:00:00 ps -ef
```



Networking:

- "-p" Port-Weiterleitung vom Host zum Container für einen dedizierten Port
- "-P" Port-Weiterleitung für alle Ports
- "--link" verknüpft zwei Container miteinander
- "--dns" DNS-Server für den Container ändern
- "--mac-address" MAC-Adresse setzen
- …



Dieses Beispiel zeigt eine einfache Netzwerkkonfiguration

```
# docker run -d -p 9000:9000 --privileged -v
/var/run/docker.sock:/var/run/docker.sock dockerui/dockerui
```

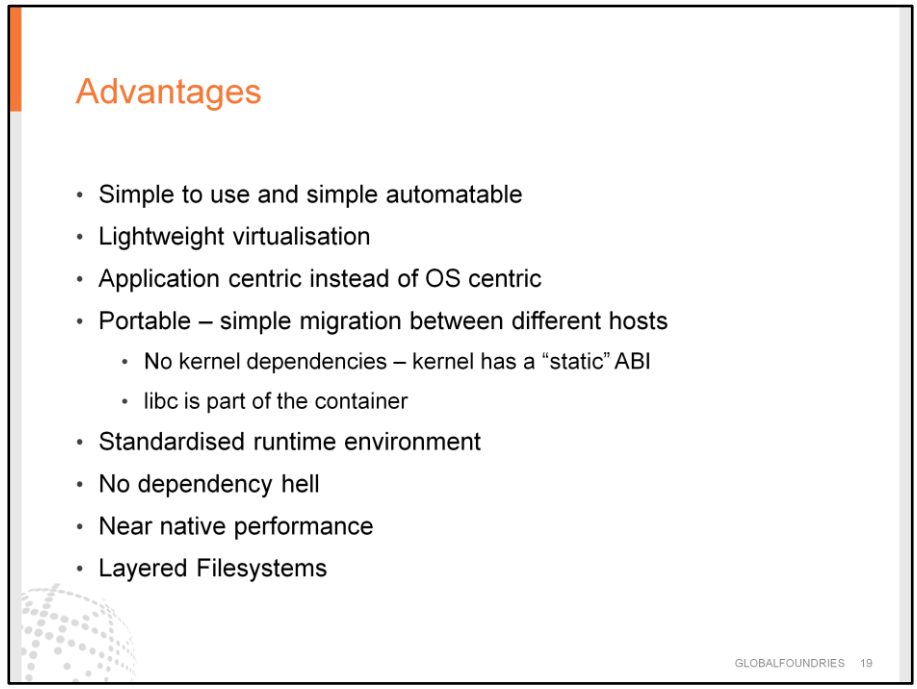
Im Browser öffnen: http://<docker host>:9000/

Options:

- "-d" Container im Hintergrund laufen lassen (detached)
- "-p <…>" Port-Weiterleitung vom Host zum Container
- "--privileged" Grant additional permissions to container
- "-v <…>" Provide host volumes to container

Agenda

1. Introduction
2. Live presentation
3. Discussion



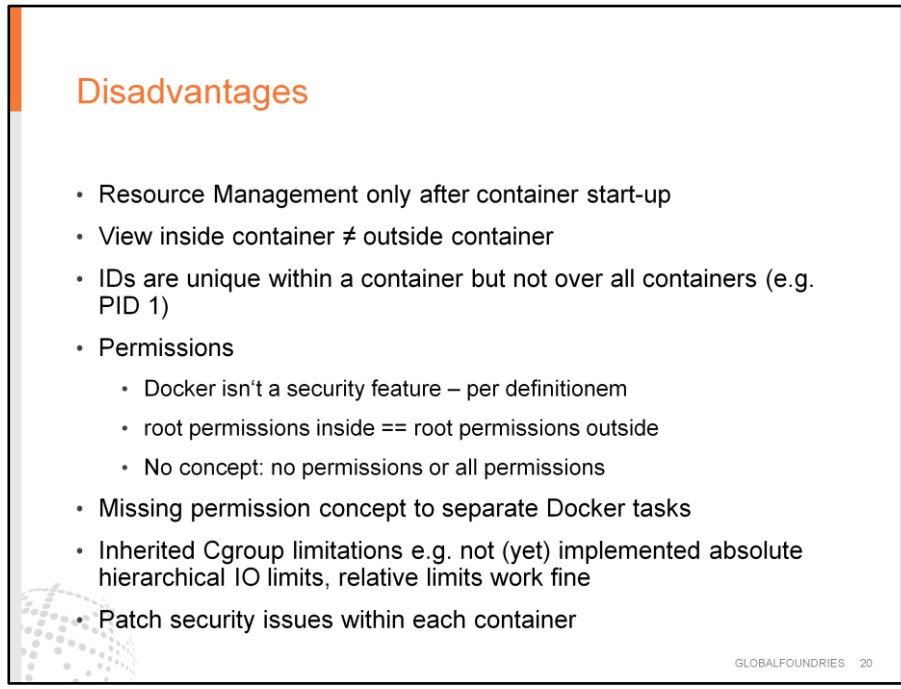
Lightweight Virtualisation:
- Unterschiede bereits im Abschnitt „Architecture“ diskutiert

Portable / Vorraussetzungen:
- Kernel-Version: 3.8 oder neuer
- Empfohlen: ab Kernel 3.10

Dependency Hell:
Keine Probleme mit verschiedenen und untereinander inkompatiblen Versionen von gemeinsam genutzten Bibliotheken

Performance:
Laut IBM (http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf)
- Docker-Performance ist immer gleich oder besser KVM
- Der Unterschied zur nativen Performance ist meßbar, aber vernachläßigbar
- Lediglich bei viel IO / auch (AUFS) muß man hinsehen

Layered Filesystems
- Die meisten anderen Container Tools haben dies nicht

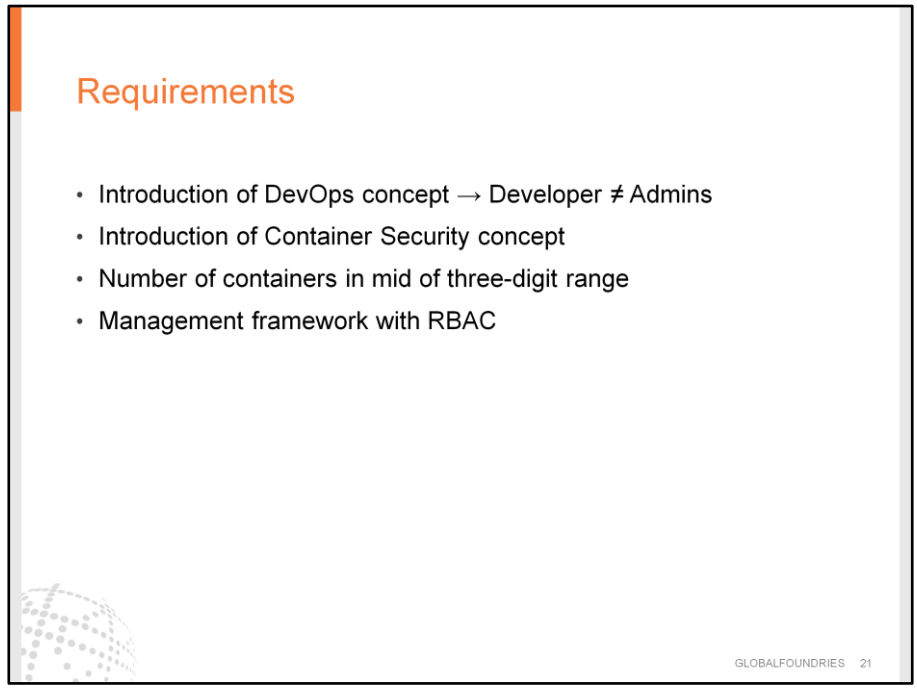


Resource Management mit Cgroups:
- ist erst möglich, nachdem der Container gestartet wurde
- vorher existieren keine Prozesse
- Race Condition zwischen schneller Nutzung der Ressourcen und dem Setzen der Limits

Innensicht ≠ Außensicht:
- IDs (PID, GID, UID, IPCID) sind nur innerhalb eines Containers eindeutig
- Innensicht: Die gleiche PID (z.B. 1) kann in mehreren Container gleichzeitig existieren
- Außensicht: Auf dem Host sind alle IDs eindeutig, haben aber nicht den selben Wert, wie innerhalb des Containers
- Sicherheitsprobleme, wenn Nutzer und Gruppen manuell in Containern angelegt werden und deren Name/ID schon an anderen Stelle verwendet wird -> Überschreiben von fremden Dateien möglich
- Änderungen an skriptbasierte Aktionen (z.B. via JSS)

Berechtigungskonzept:
- Kein RBAC
- Alles oder nichts -> root oder Mitglieder der Gruppe docker dürfen alles
- Mitglieder der Gruppe docker können sich auf Umwegen über einen eigenen Container Root-Rechte auf dem Host beschaffen z.B. Root-Shell oder /etc/passwd, /etc/shadow manipulieren „docker run –v /etc/shadow:/myshadow“ & vi /myshadow
- DevOps-Konzept: Entwickler haben auch root-Rechte

Sonstiges:
- Anwendungen sind sich häufig nicht Einschränkung ihrer Ressourcen „bewusst“



DevOps:
- Zuständigkeiten klären -> DevOps Konzept einführen?
- Entwickler sind keine Admins und Admins sind keine Entwickler

Container Security Concept:
- Handhabung von verschiedenen Berechtigungen
- Compliance Check des Containers
- Prozedur für Security Updates

Open – to discuss

- Minimalised OS for Docker hosts like Project Atomic
- Using SELinux
- Alternatives: lmctfy, LXC, systemd-nspawn, …
- CaaS
- Backup / Recovery
- …

Live Migration via CRIU (checkpoint/restore in userspace) und dem P.HAUL Project

Thank you

GLOBALFOUNDRIES®