Private Nutzerumgebungen mit dynamisch zugeordneten Ressourcen in Linux-Containern

Dresden, 23.03.2017

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Gliederung

1) Aufgabenstellung

- Dynamische Verwaltung sowie flexibilisierte Zuweisung von potentiell knapp verfügbaren Device-Ressourcen in einem System mit mehreren Benutzern
- Benutzerumgebung mit weitgehenden Verwaltungsrechten
- Möglichst sichere Abgrenzung von Host System und Benutzerumgebung durch unprivilegierte Container
- Nutzer soll eine virtuell stabile Systemsicht erhalten
- Systemzugriff über SSH

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

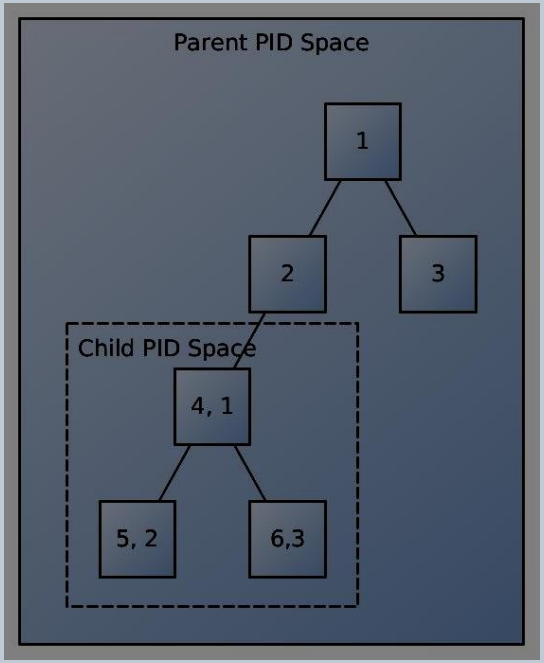
2.1) Container, Aufbau

- Virtualisierungsansatz auf Betriebssystemebene zur Isolation von Prozessen und Prozessgruppen
- Nutzt Kerneltools zur Organisation und Isolation
- Wichtigste Tools: namespaces und cgroups
- Container können auf dem Host System root Rechte (privileged) oder user Rechte (unprivileged) haben
- Ziel: leichtgewichtige, skalierbare und sichere Benutzerumgebung verschiedener Anwendungen auf einem Host

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

2.1) Container, Aufbau

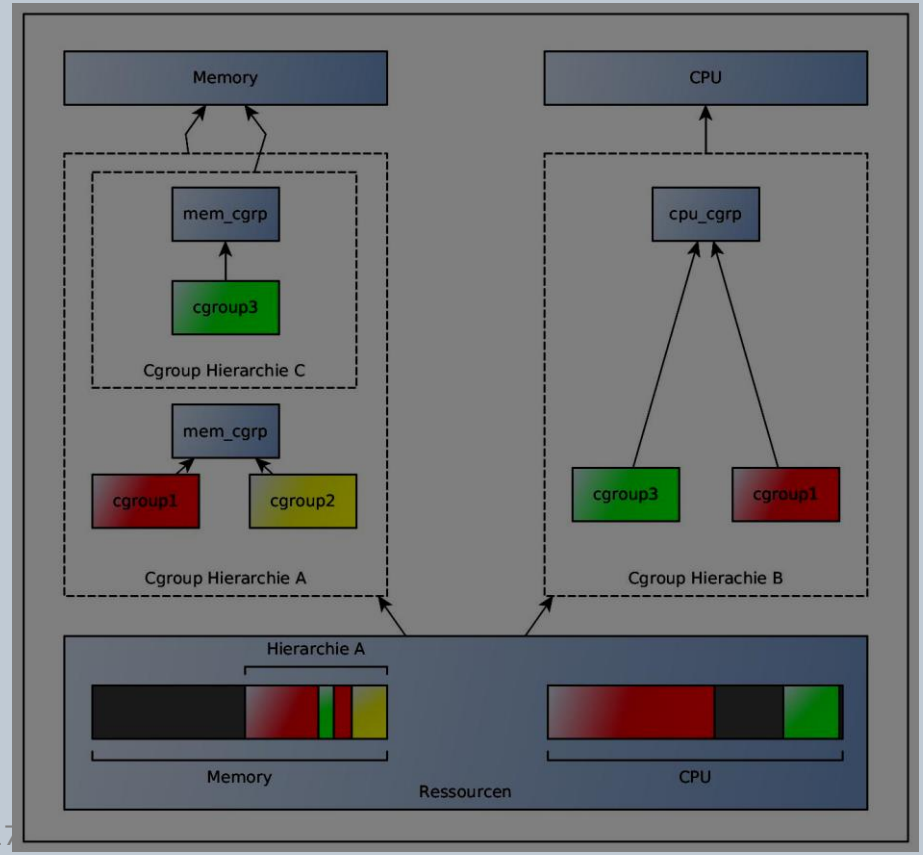
- Namespaces:
- Managementsystem zur Rechteverwaltung
- sowie zur Isolation
- Ansatz: ID besitzt zwei Werte, einen Host und einen
- User Wert
- Resultat: Rechteverwaltung eines Nutzers kann
- losgelöst vom Host organisiert werden
- Routing von Datenpaketen kann auf Hostseite
- durchgeführt werden, Nutzer kann Schnittstellen
- wie gewohnt nutzen
- Nutzung: virtual Ethernet, root Rechte innerhalb eines
- Containers aber user Rechte auf dem Host System



DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

2.1) Container, Aufbau

- Control Groups:
- Ressourcenmanagement über eine Ordnerstruktur mit editierbaren Textdateien
- Regelt: blkio, cpu, cpuacct, cpuset, devices, freezer, memory, net_cls, net_prio, ns
-

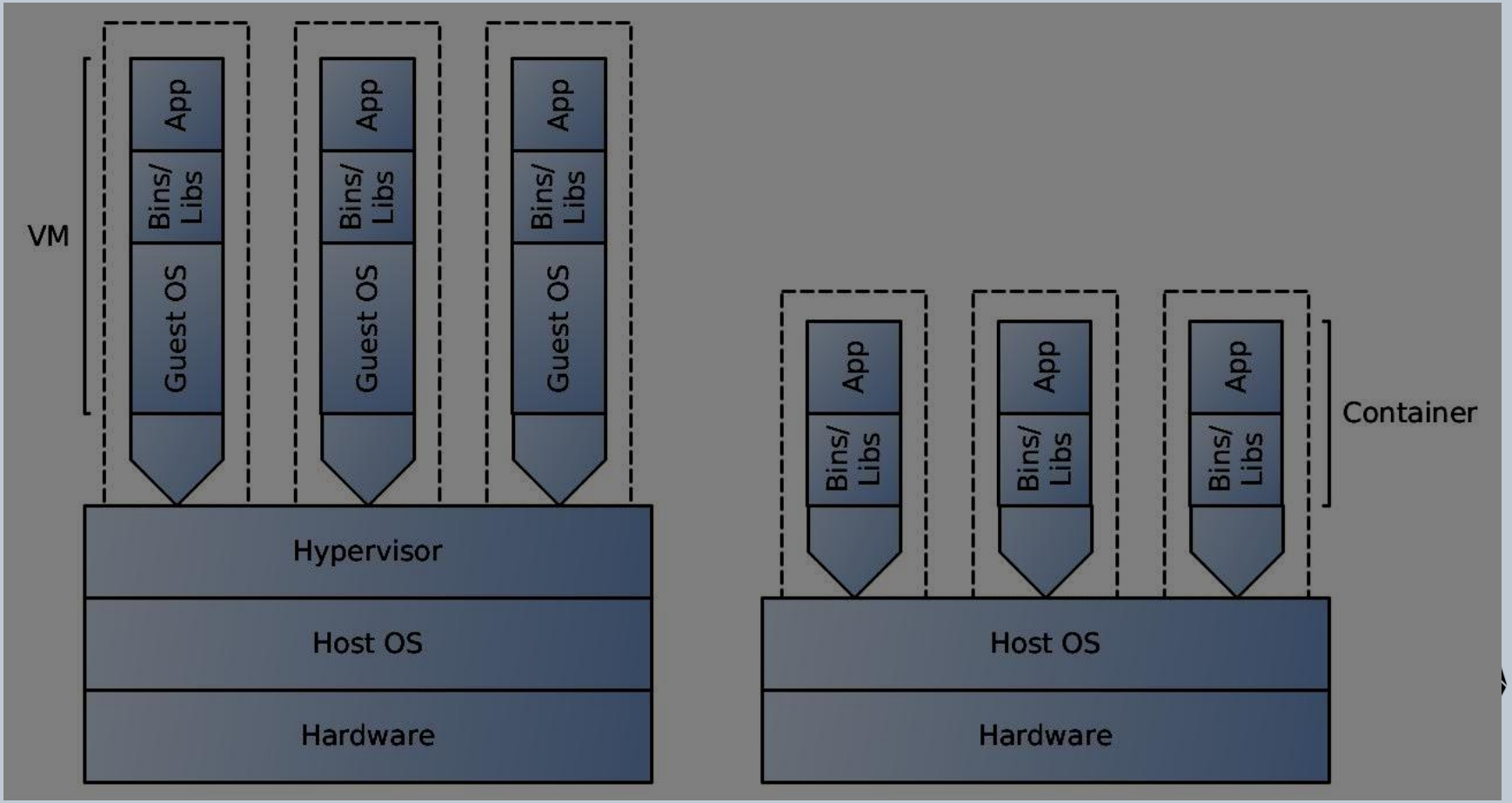


DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

2.2) Container, Vorteile und Nutzen

- Vielfältige Anwendungsmöglichkeiten: Cloud Systeme, Testumgebungen zur Programmentwicklung, Micro Server, Benutzerverwaltung
- Sehr effiziente Hardwarenutzung sowie -verwaltung
- Start Zeiten entsprechen Prozessstartzeiten (~µs)
- Feine Strukturierung durch Anwendungscontainer möglich
- User System und Host System müssen (meistens) gleichen Kernel benutzen
- Aktuell üblicheres Konzept: Virtual Machine
- Höhere Isolation, flexiblere Systemwahl
- Wesentlich mehr Hardwareaufwand, weniger dynamisch

2.3) Container, Vorteile und Nutzen



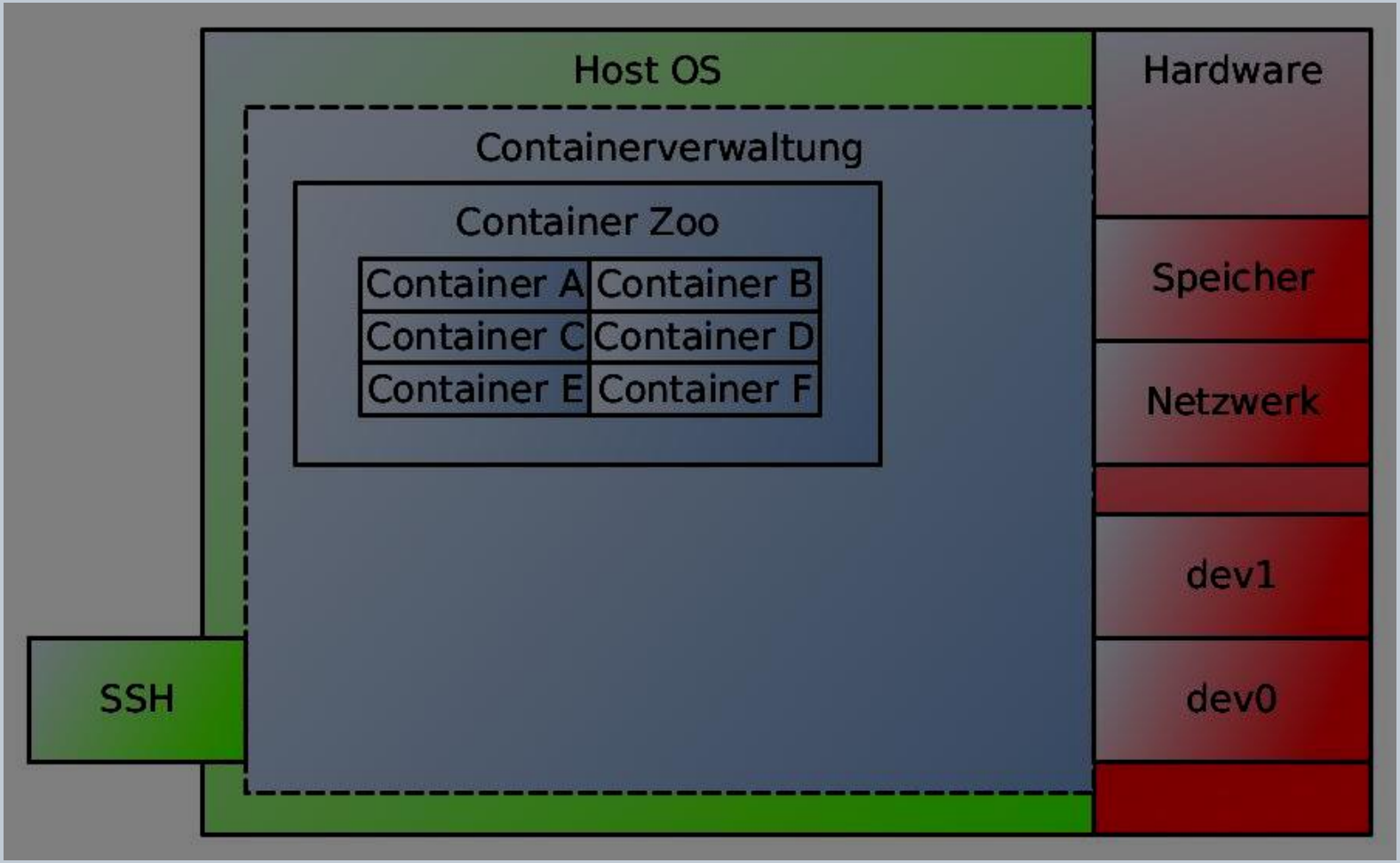
2.4) Container, Linux Container (LXC)

- Commandline-Benutzerinterface zur Organisation von Containern
- Aktuell in Version 2.0 frei verfügbar
- Organisiert erstellen, starten, stoppen, beenden sowie zerstören von Containern, inklusive der jeweiligen Schnittstelle zum Hostsystem
- Stellt Tools zur Wartung sowie Betriebssystem Templates zum erstellen von Containern zur Verfügung
- Verwaltet Konfigurationsdateien

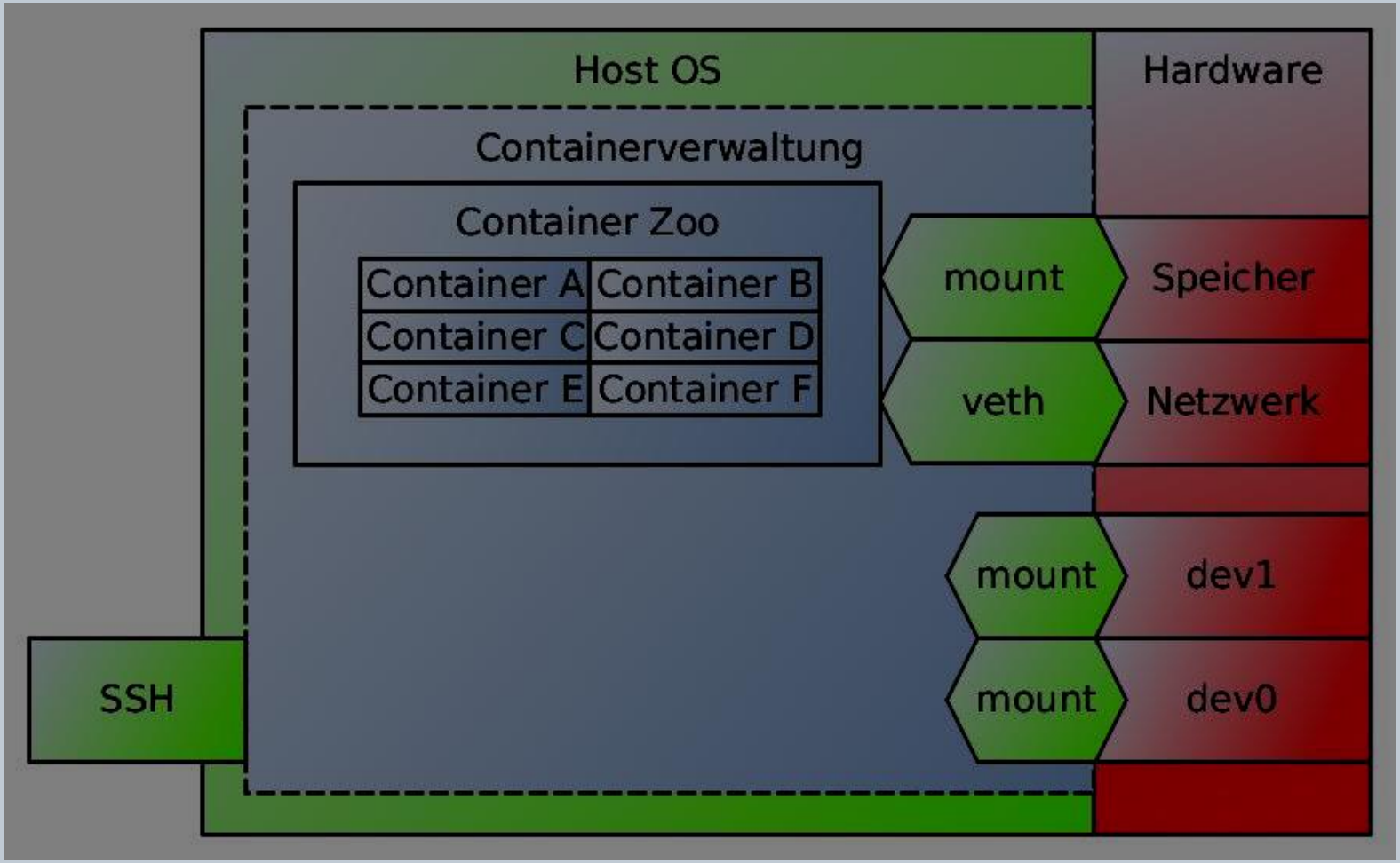
2.5) Container, Konfiguration

- Konfiguration findet über zwei Vorlagen sowie eine editierbare Textdatei statt
- Standardkonfiguration beinhaltet uid/ gid mapping, rootfs Verzeichnispfad, Containername und anlegen der virtuellen Netzwerkschnittstelle
- Weitere Einstellungen, abhängig von privileged/ unprivileged möglich
- Einrichten eines persistenten Verzeichnisses über einen mount Befehl
- Dateien und Verzeichnisse werden über einen mount Befehl dem Container zugänglich gemacht

3.1) Systemanforderungen, Konzept

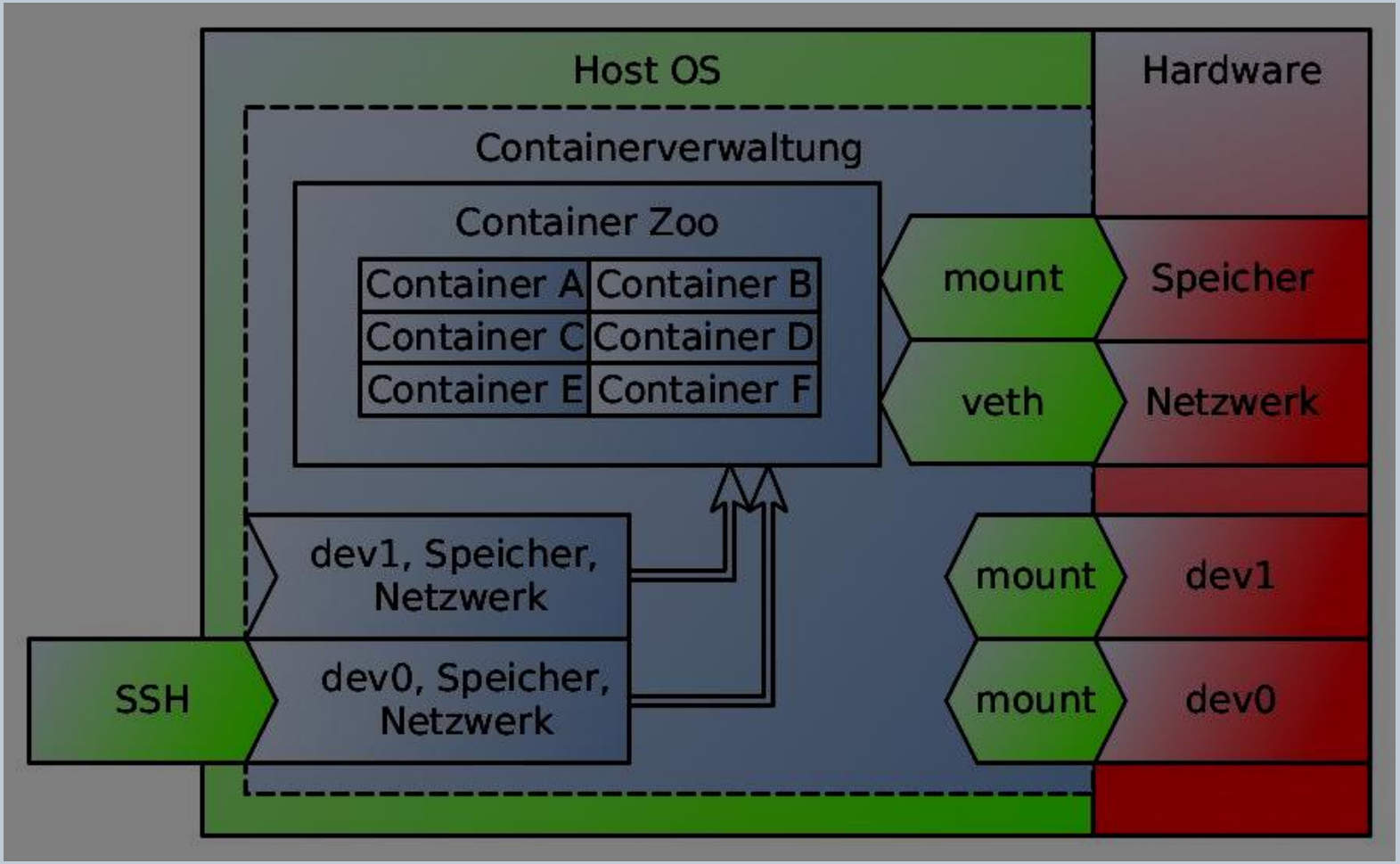


3.1) Systemanforderungen, Konzept



DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

3.1) Systemanforderungen, Konzept



DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

3.2) Systemanforderungen, Secure Shell

- Systemzugriff erfolgt über SSH
- Nutzer muss einen Schlüssel generieren und den öffentlichen Teil auf das remote System Kopieren
- Eindeutige Identifikation über den Schlüssel ermöglicht Individualisierung über *authorized_keys* Datei
- Jedem Schlüssel wird eigene *$USER* Umgebungsvariable zugeordnet
- Verbindung mit diesem Schlüssel löst Befehl aus, Skript wird gestartet
- Ende des Skriptes bedeutet beenden der Verbindung

3.3) Systemanforderungen, Benutzerrechte

- Auf dem Host System müssen UID und GID des Containers existieren
- Ordnerstrukturen des Hosts müssen angelegt sein und mit den entsprechenden Rechten versehen sein
- Devicefiles müssen durch Access Control Lists für den Container zugänglich gemacht werden

4) Skriptablauf

Aktion	Kommentar
SSH login	Schlüssel identifizieren, $USER setzen, Skript starten
Device Verfügbarkeit prüfen	Containerkonfiguration durch Kommentar, mount für devicefile oder Verzeichnis ergänzen
Container starten	
Zuweisen der Benutzerkonsole	System nun frei benutzbar, Skript pausiert
Herunterfahren des Containers	Skript läuft weiter
Korrigieren der Containerkonfiguration	Device über Kommentar identifizieren, Zeilen entfernen

5) Abschlussbetrachtung und Ausblick

- Einfaches, in wenigen Schritten erweiterbares System
- Host System durch unprivilegierte Container geschützt, Benutzer hat Zugriff auf vollständiges Betriebssystem
- Systemodifikationen auf Hostseite sind aufwändig, jeder Benutzer muss angepasst werden
- Mögliche Variationen:
- Ein statischer immer identischer Container je Device Ressource, Individualisierung durch eigenes /home Verzeichnis
- Anwendungscontainer, entweder je Device Ressource oder je Nutzer, Individualisierung durch eigenes Arbeitsverzeichnis zur Projektablage

6) Quellen

•Vortrag

•Rose, Rami: „Namespaces and cgroups, the basis of Linux containers“. Stand 02.2016 http://www.netdevconf.org/1.1/proceedings/slides/rosen-namespaces-cgroups-lxc.pdf (18.03.2017) [Vortrag]

•https://en.wikipedia.org/wiki/Operating-system-level_virtualization (18.03.2017)

•https://wiki.ubuntuusers.de/LXC/ (18.03.2017)

•https://en.wikipedia.org/wiki/Linux_containers (18.03.2017)

•https://de.wikipedia.org/wiki/Secure_Shell (18.03.2017)

•Graber, Stephane: „So what‘s LXC?“. Stand 20.12.2013https://stgraber.org/2013/12/20/lxc-1-0-your-first-ubuntu-container/ (19.03.2017) [Blogeintrag]

•https://www.redhat.com/en/containers/whats-a-linux-container (19.03.2017)

•https://opensource.com/resources/what-are-linux-containers (19.03.2017)

•Wang, Chenxi: „Containers 101: Linux containers and Docker explained“. Stand 26.05.2016http://www.infoworld.com/article/3072929/linux/containers-101-linux-containers-and-docker-explained.html (19.03.2016)

•Kerrisk, Michael: „User namespaces progress“. Stand 13.12.2012 https://lwn.net/Articles/528078/ (19.03.2017)

•https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html (19.03.2017)

•Ridwan, Mahmud: „Separation Anxiety: A Tutorial for Isolating Your System with Linux Namespaces“. https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces (19.03.2017)

•Skript

•Geschke, Dirk: „LinuXContainer, Nutzung unprivilegierter LinuXContainer als normaler Nutzer“. Stand 20.01.2015 http://www.lug-erding.de/vortrag/LXC.pdf (26.02.2017) [Vortrag]

•https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/sec-devices.html (26.02.2017)

•Schlittermann: „SSH ohne Passwort – Kurze Anleitung zur Nutzung“. Stand 15.05.2014 http://www.schlittermann.de/doc/ssh.html (12.03.2017)

•https://github.com/lxc/lxc/issues/1291 (26.02.2017) [Forum, Diskussion]

•https://wiki.ubuntuusers.de/ACL/ (12.03.2017)

•https://linuxcontainers.org/ (19.03.2017)

•O‘Rilley & Associates: „Running Linux, third edition“. Chapter: „Device Files“. http://docstore.mik.ua/orelly/linux/run/ch06_03.htm (26.02.2017)

•Graber Stephane:LXC 1.0 Security features“. Stand 01.01.2014 https://stgraber.org/2014/01/01/lxc-1-0-security-features/ (26.02.2017) [Blogeintrag]

•https://superuser.com/questions/48783/how-can-i-pass-an-environment-variable-through-an-ssh-command (12.03.2017) [Forum, Diskussion]

»Wissen schafft Brücken.«