Modul CNT
Container Standards und Technologien

September 2021

Marcel Bernet

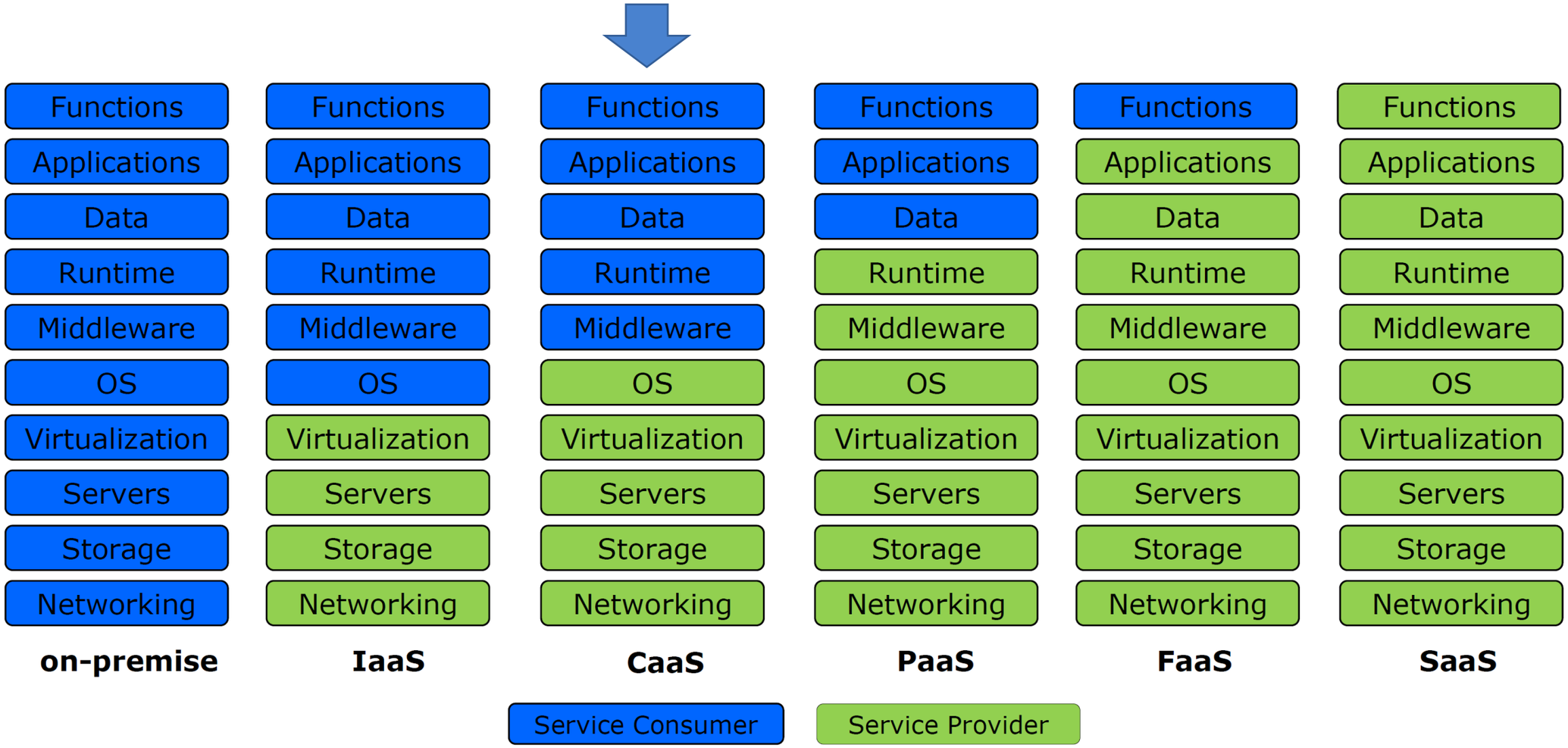
Lernziele

★ Sie haben einen Einblick in die Container Basics.

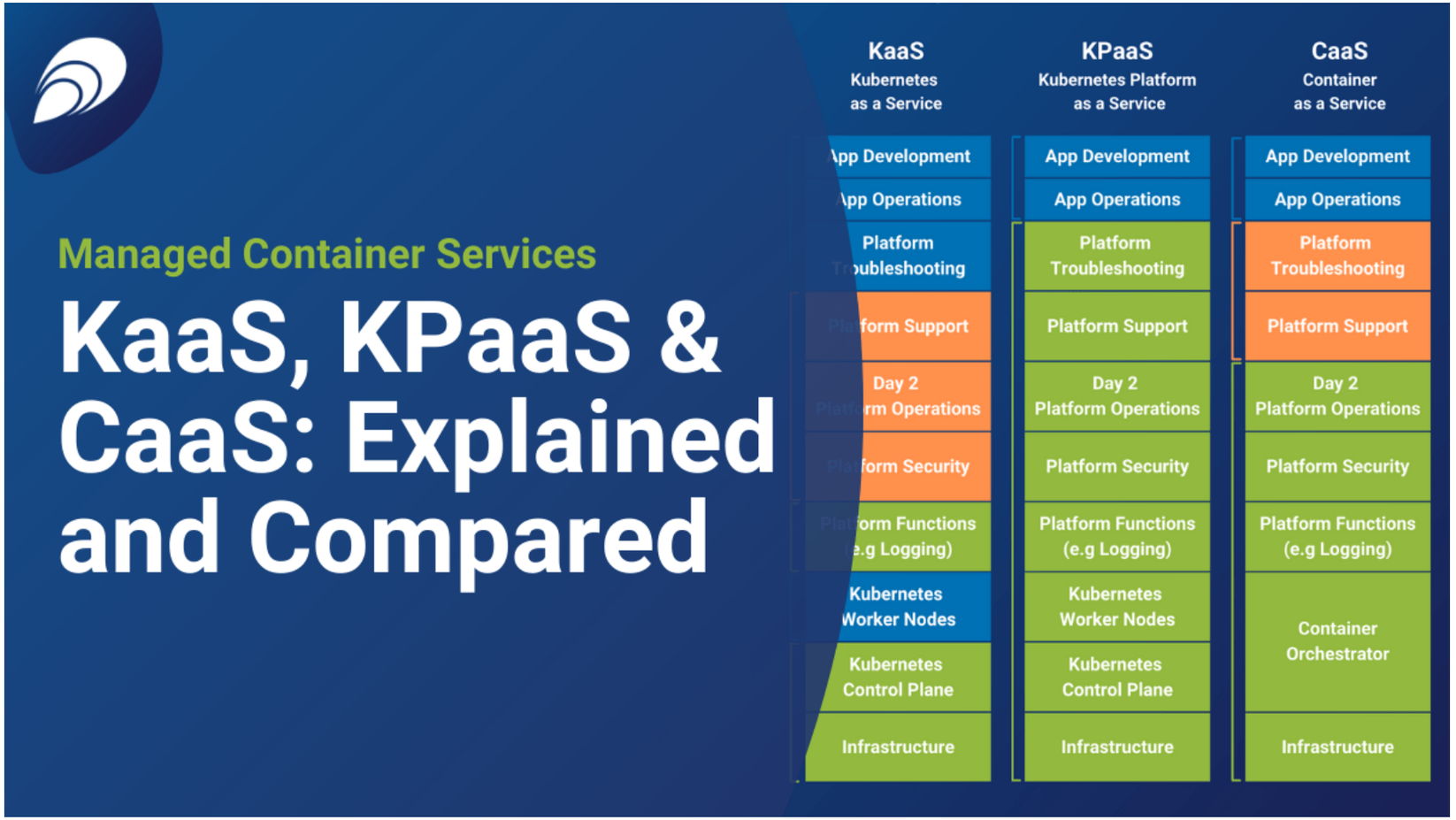
Zeitlicher Ablauf

- ★ Cloud: Day 2 (Cloud Native) - Cloud computing stack
- ★ Von der traditionellen Informatik, zu Virtuellen Maschinen und Containern
- ★ Container-Geschichte
- ★ Linux Namespaces und Container
- ★ Hands-on
- ★ Reflexion
- ★ Lernzielkontrolle

Cloud: Day 2 (Cloud Native) - Cloud computing stack

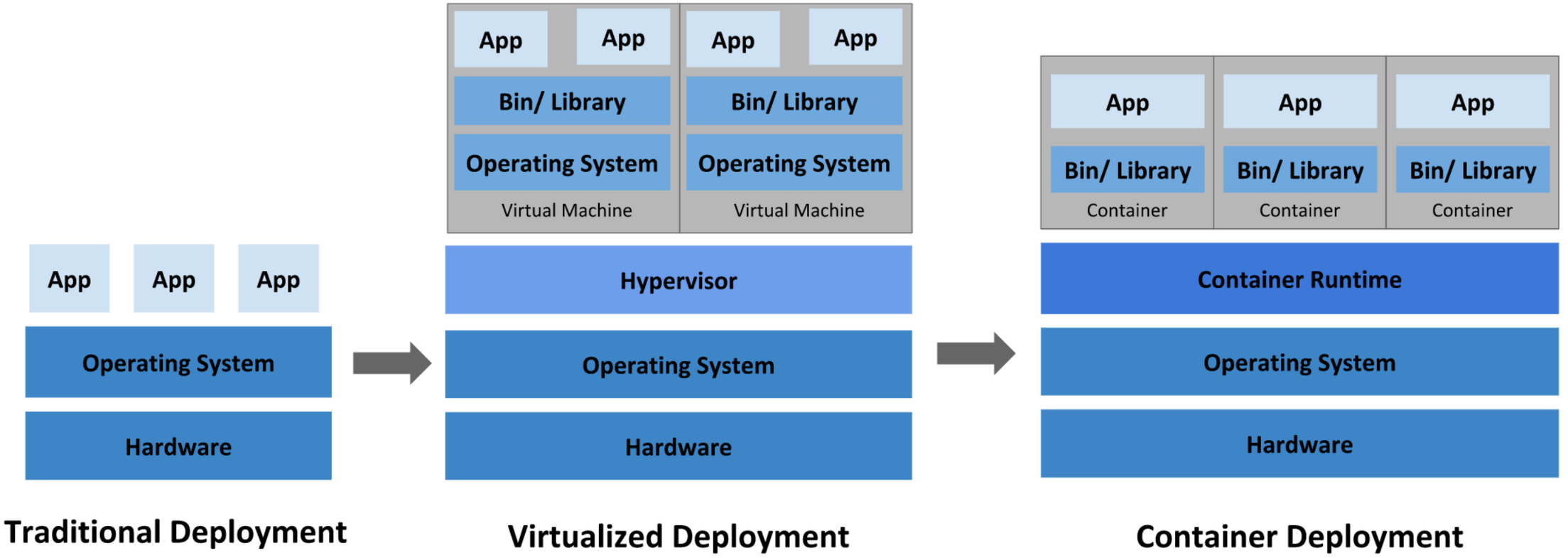


Managed Container Services: KaaS, KPaaS & CaaS Explained and Compared



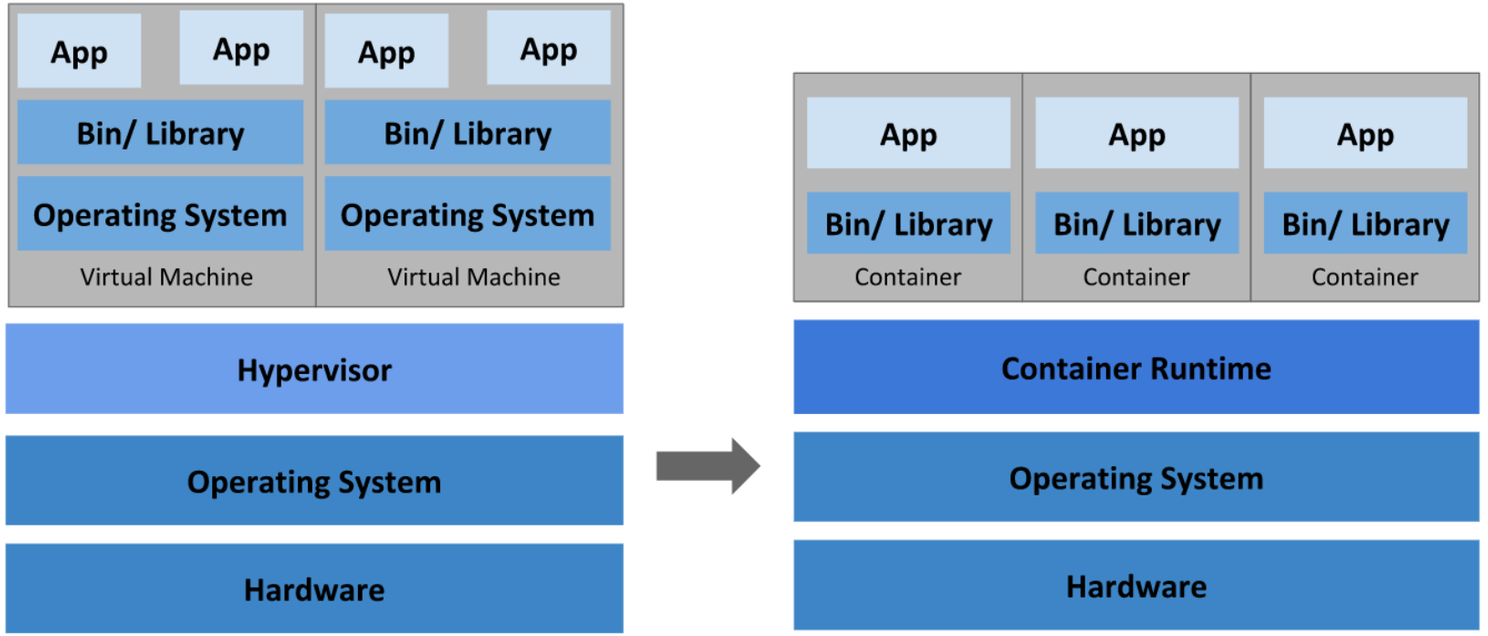
https://elastisys.com/managed-container-services-kaas-kpaas-caas-explained-and-compared/

Von der traditionellen Informatik, zu Virtuellen Maschinen und Containern



Quelle: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Virtuelle Maschinen (schwergewichtig) vs. Container (leichtgewichtig)



Virtualized Deployment

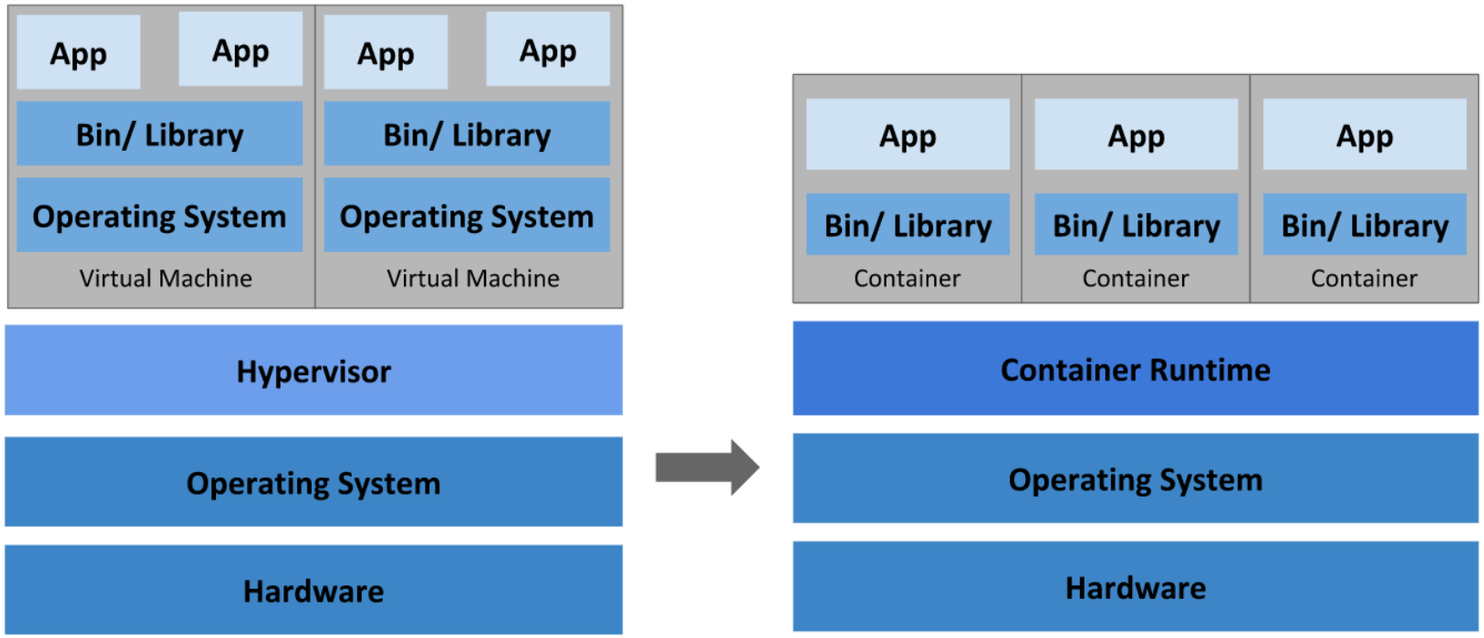
- Hypervisor
- Guest OS
- Applikation und ihre Abhängigen z.B. Libraries / DLLs in VM
- Schwergewichtig (Platzbedarf: GB)

Container Deployment

- Container Engine
- Applikation (Microservice) und ihre Abhängigen z.B. Libraries / DLLs im Container
- Leichtgewichtig (Platzbedarf: KB - MB)

Quelle: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Virtuelle Maschinen (manuelle Installation) vs. Container (Images - Paketiert)



Virtualized Deployment

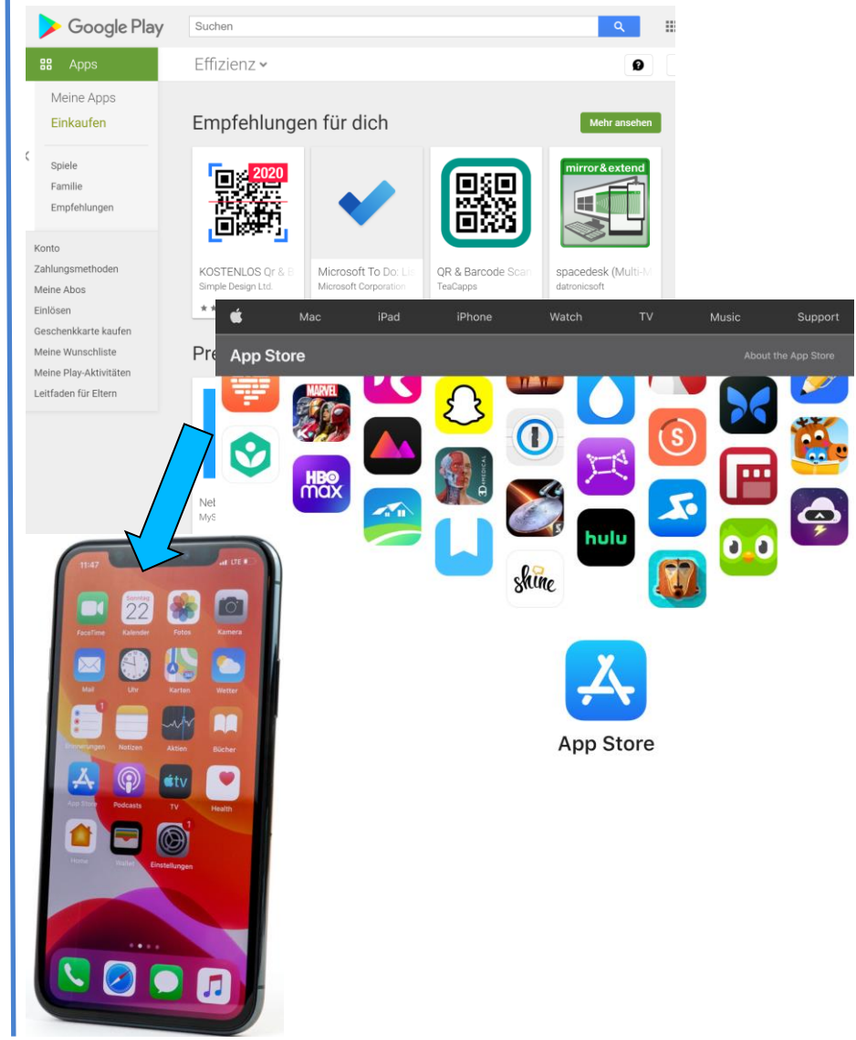
- ISO-Image (CD-Rom) und Manuelle Installation von Applikationen oder Installations-Script z.B. Cloud-init, Ansible, Chef, Puppet, Vagrant etc.

Container Deployment

- (Automatische) Bereitstellung der Applikationen (Microservices) als Container Images
- Paketiert mittels einer einheitlichen Beschreibungsdatei (Dockerfile)
- Images abgelegt in Registries

Quelle: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Apps (Smartphone) Ökosystem



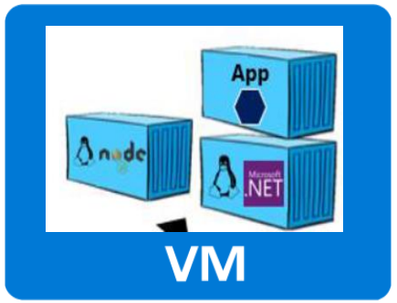
Brauchen wir noch Virtuelle Maschinen? Oder welche Einschränkungen haben Container?

★ (Schlechtes) Beispiel: Container mit systemd, Datenbank und Web Server (PHP)

- **Beispiel verletzt Regel: ein Container ein Prozess**

```
FROM registry.access.redhat.com/ubi8/ubi-init
MAINTAINER fatherlinux &lt;scott.mccarty@gmail.com&gt;
RUN yum install -y mariadb-server mariadb php php-apcu php-intl
php-mbstring php-xml php-json php-mysqlnd crontabs cronie iputils
net-tools;yum clean all
RUN systemctl enable mariadb
RUN systemctl enable httpd
RUN systemctl disable systemd-update-utmp.service
ENTRYPOINT ["/sbin/init"]
CMD ["/sbin/init"]
```

★ **JA!**, für Legacy Applikationen und als Worker Node (Behälter) für Container.



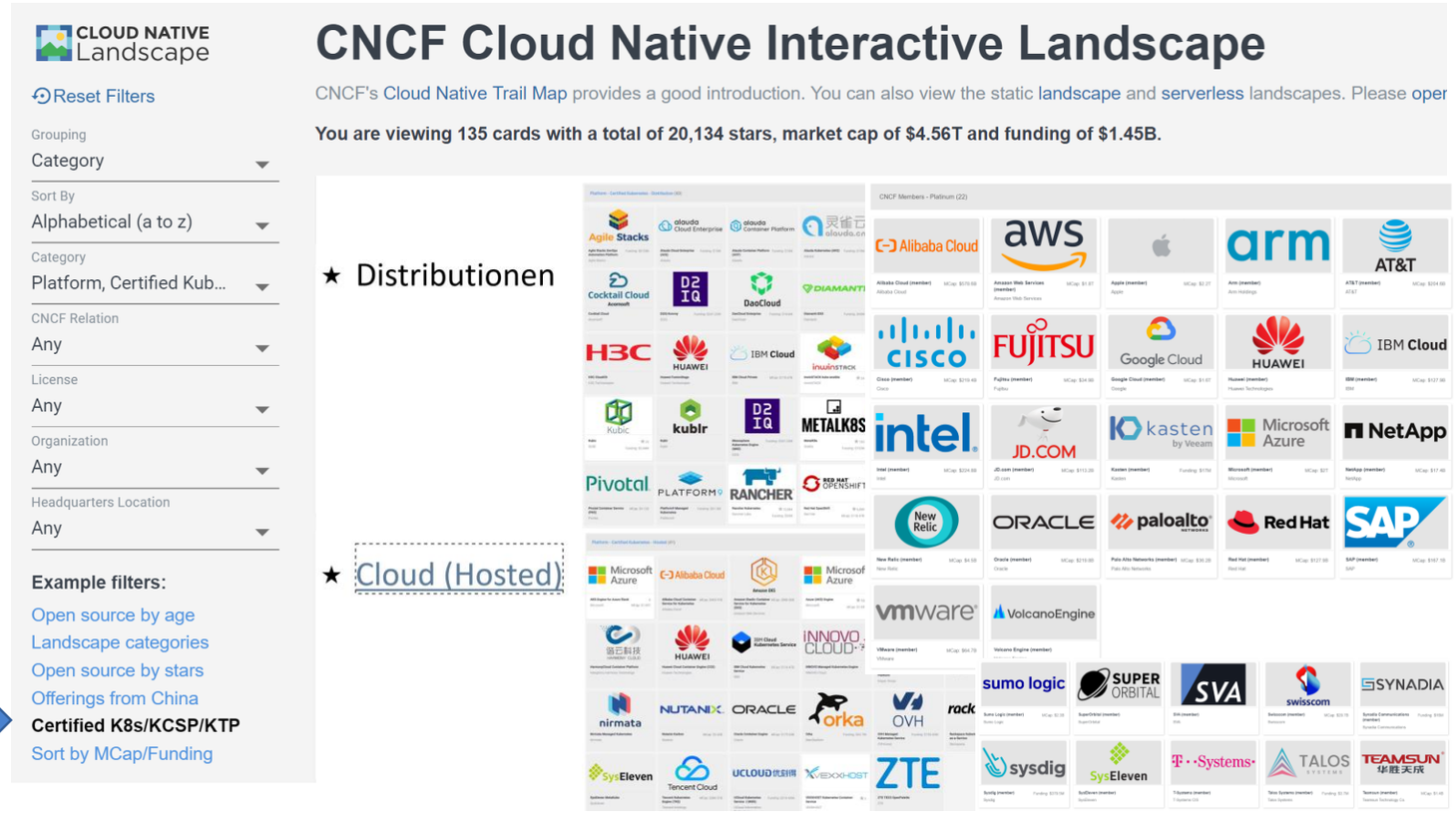
Quellen: https://www.redhat.com/sysadmin/wordpress-container
http://crunchtools.com/moving-linux-services-to-containers/

Container - Geschichte

- Container sind ein altes Konzept. Schon seit Jahrzehnten gibt es in UNIX-Systemen den Befehl **chroot (1979)**, der eine einfache Form der Dateisystem-Isolation bietet.
- Seit 1998 gibt es in FreeBSD das Jail-Tool, welches das chroot-Sandboxing auf Prozesse erweitert.
- **Solaris** Zones boten **2001** eine recht **vollständige Technologie** zum Containerisieren, aber diese war auf Solaris OS beschränkt.
- Ebenfalls 2001 veröffentlichte Parallels Inc. (damals noch SWsoft) die kommerzielle Containertechnologie Virtuozzo für Linux, deren Kern später (im Jahr 2005) als Open Source unter dem Namen OpenVZ bereitgestellt wurde.
- Dann startete **Google** die Entwicklung von **CGroups** (Beschränkungen für Prozesse, Memory, CPU) für den Linux-Kernel und begann damit, seine **Infrastruktur in Container zu verlagern**.
- Das Linux Containers Project (LXC) wurde **2008** initiiert, und in ihm wurden (unter anderem) **CGroups, Namespaces und die chroot-Technologie zusammengeführt**, um eine vollständige Containerisierungslösung zu bieten.
- **2013** lieferte **Docker** schließlich die fehlenden Teile für das Containerisierungspuzzle, und die Technologie begann, den Mainstream zu erreichen.
- **Heute**: Kubernetes hat die Lücke zu Orchestrierung (vereinfacht: Management) von Container Umgebungen geschlossen. Reine Container Umgebungen, wie Docker, verlieren an Bedeutung.

Übung: Wie viele Kubernetes Distributionen und Hosted Kubernetes Umgebungen gibt es?

★ Öffnet die CNCF Landscape und sucht alle Kubernetes (K8s) zertifizierten Distributionen und Hosted Umgebungen:



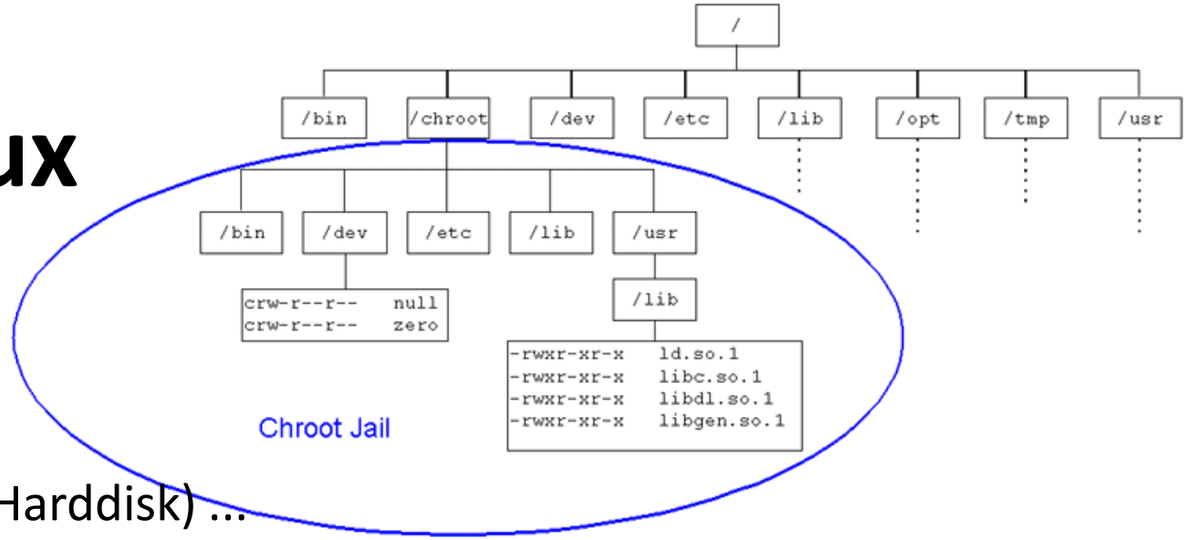
vs.

Docker CE - **Zukunft ungewiss?**
Docker EE - **verkauft an Mirantis**
Kubernetes is deprecating Docker
(Ende Unterstützung **31.12.21**)

Container basieren auf Linux Konzepten (= Linux Namespaces)

- ★ **Aufgabe Linux Namespaces**
  - Ressourcen des Kernsystems (in diesem Falle also des Kernels) voneinander zu isolieren
- ★ **Arten von Namespaces**
  - IPC -Interprozess-Kommunikation
  - NET - Netzwerkressourcen
  - PID -Prozess-IDs
  - USER - Benutzer/Gruppen-Ids
  - (UTS - Systemidentifikation): Über diesen Namespace kann jeder Container einen eigenen Host- und Domänennamen erhalten.
- ★ **Weitere Informationen**
  - https://docs.docker.com/engine/security/seccomp/
  - https://kubernetes.io/blog/2021/08/25/seccomp-default/ - Kubernetes 1.22
  - https://medium.com/faun/the-missing-introduction-to-containerization-de1fbb73efc5

Container basieren auf Linux Konzepten (vereinfacht)



★ **Dateisystem (ls -l /)**

- Alles ist eine Datei z.B. /proc/cpuinfo, /dev/sdaX (Harddisk) ...
- Es gibt keine Laufwerke alles ist via / (root) erreichbar. Mehrere Roots sind möglich.
- Die Unterstützung mehrere Filesysteme (ext4, overlay2, aufs …) ist die Regel und nicht die Ausnahme.

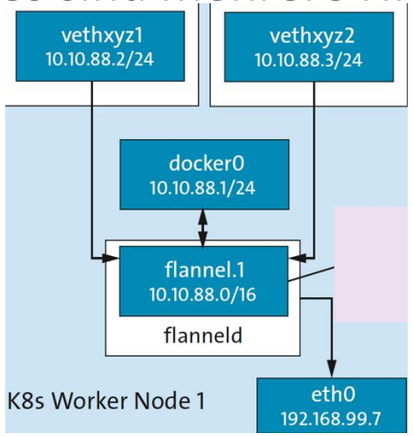
★ **Prozesse (pstree -n -p)**

- Prozesse sind hierarchisch angeordnet
- Der erste Prozess hat die ID 1, es sind mehrere Hierarchien möglich.

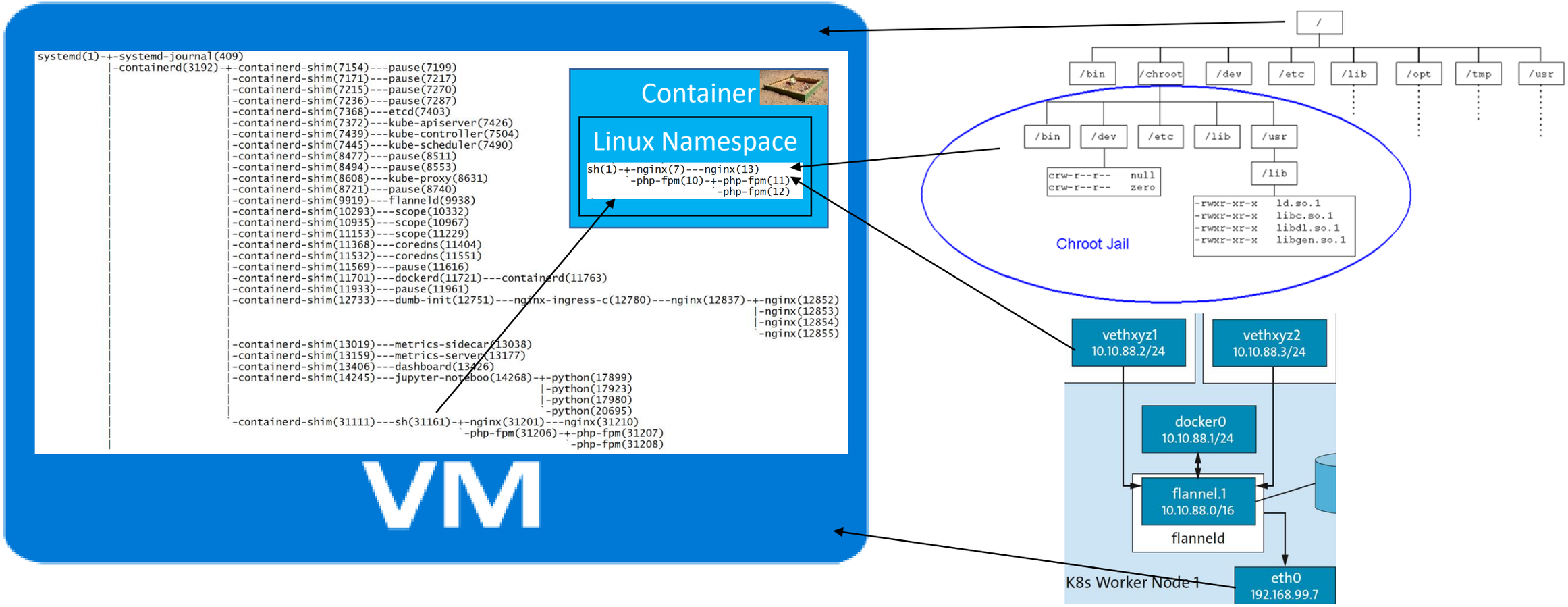
```
systemd(1)-+-accounts-daemon(638)
           |-containerd(3867)-+-containerd-shim(7978)---pause(8076)
           |                  |-containerd-shim(15970)---metrics-sidecar(15987)
           |                  |-containerd-shim(16118)---metrics-server(16133)
           |                  `-containerd-shim(747040)---sh(747058)-+-nginx(747102)---nginx(747108)
           |                                                         `-php-fpm(747105)-+-php-fpm(747106)
           |                                                                           `-php-fpm(747107)
```

```
sh(1)-+-nginx(7)---nginx(13)
      `-php-fpm(10)-+-php-fpm(11)
                    `-php-fpm(12)
```

★ Es sind **Subnetzwerke** möglich.

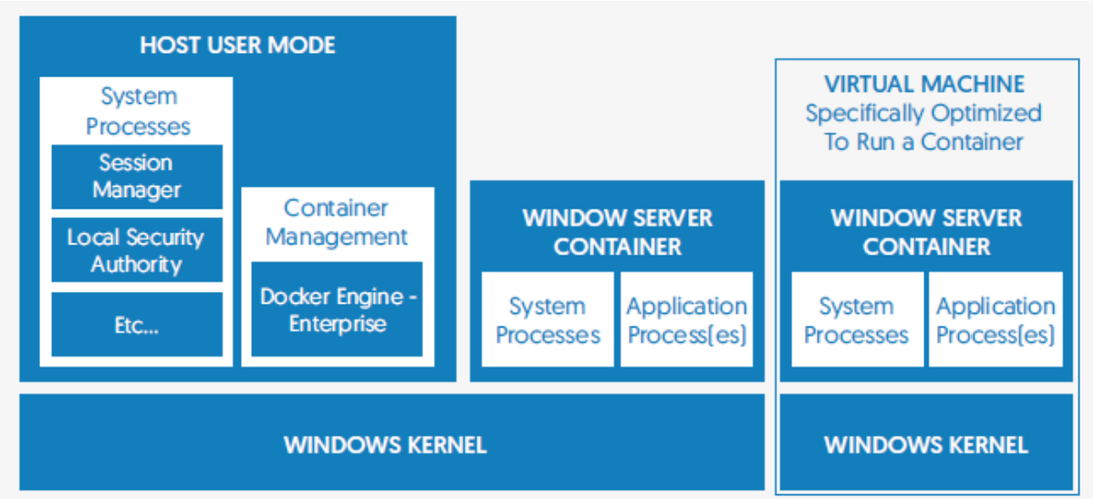


Virtuelle Maschine vs. Container/Namespace



Siehe auch: [How Docker Works - Intro to Namespaces]

Was ist mit Windows? Windows Container?



- **Host User Mode:** Analog User Mode auf Windows Server
- **Windows Server-Container**: Bieten Anwendungsisolation mithilfe einer Technologie zum Isolieren von Prozessen und Namespaces
- **Hyper-V-Isolierung**: Erweitert die von Windows Server-Containern bereitgestellte Isolierung, indem jeder Container in einem hochgradig optimierten virtuellen Computer ausgeführt wird.
- Quelle: https://docs.microsoft.com/de-ch/virtualization/windowscontainers/about/index#windows-container-types
- Azure Windows Nodes: https://docs.microsoft.com/en-us/azure/aks/windows-container-cli

Reflexion

- ★ Container sind ein altes Konzept.
- ★ Sie basieren, vereinfacht, auf Linux Namespaces.
- ★ Bei Day 2 (Cloud native) treten Infrastrukturen wie VMs und Netzwerke in den Hintergrund und abstrahierte Technologien wie Container und Serverless (Function as a Service) in den Vordergrund.

Lernzielkontrolle

★ Sie haben einen Einblick in die Container Basics.

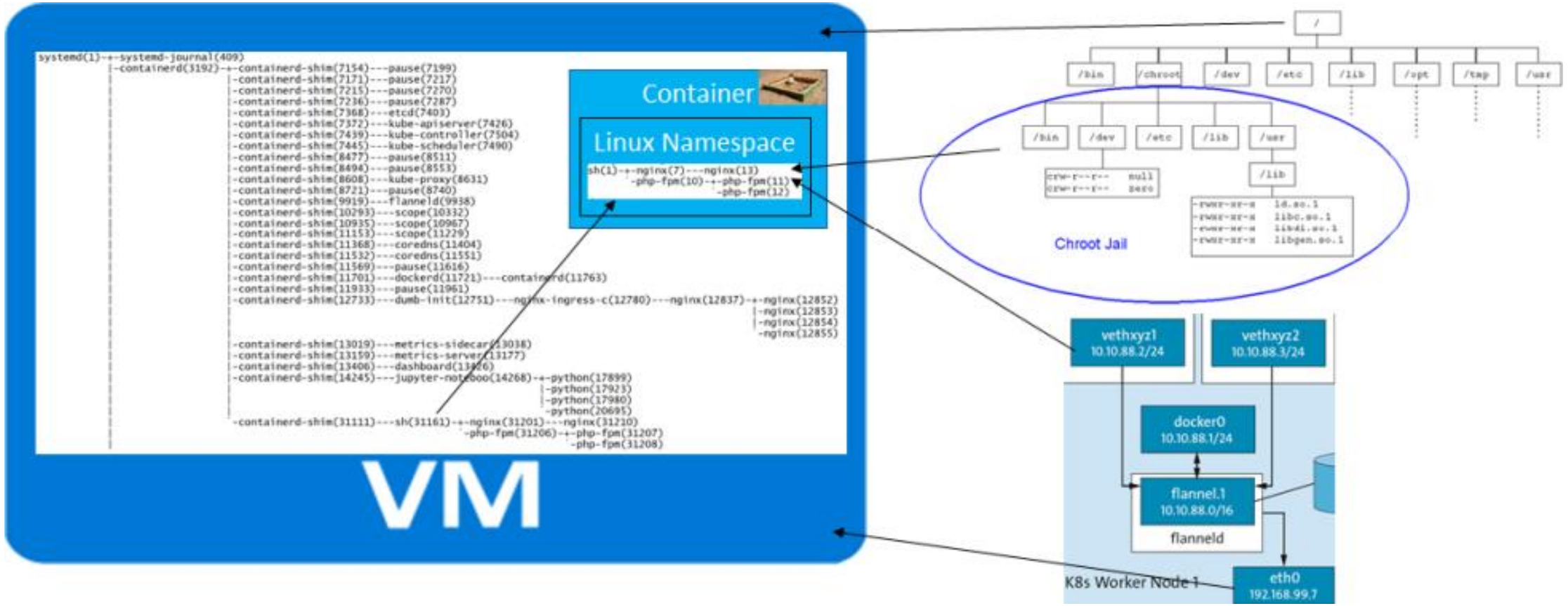
Selbstcheck

★ Ich kann mich mit einer Linux Maschine via SSH verbinden, mich auf dem Betriebssystem (Linux) zurechtfinden und Dateien editieren (z.B. mittels nano, vi oder Remote via Bitvise).

- **NEIN**: nächstes Hands-on auslassen

★ **JA**: Hands-on in der MAAS Cloud mit Cloud-init Scripten durchspielen.

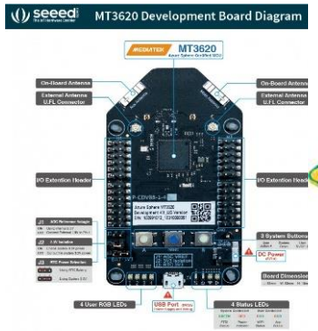
Hands-on: Container Standards und Technologien



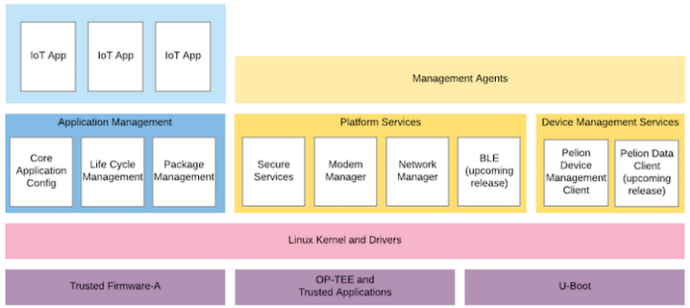
https://gitlab.com/ch-tbz-hf/Stud/cnt/-/tree/main/2_Unterrichtsressourcen/H#hands-on

Linux Container laufen auf verschiedenen Prozessor Architekturen mit Linux (MCU, CPU, GPU)

IoT

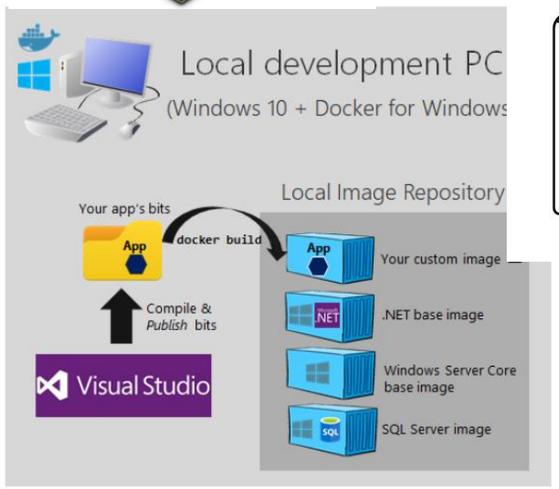


Edge



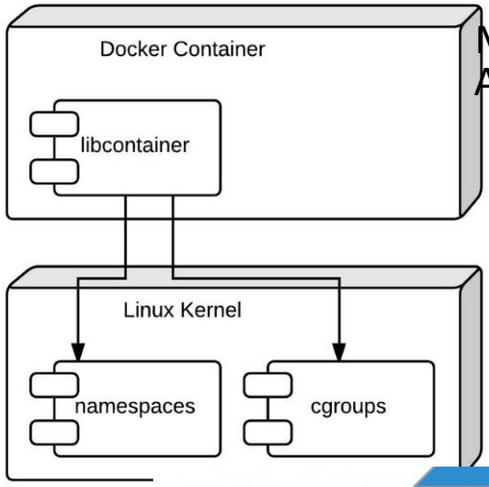
Arm Mbed + Raspian Linux OS

ARM Cortex A MCU



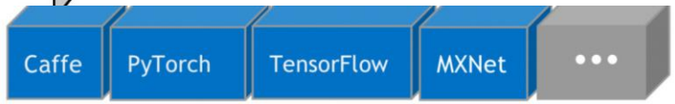
PC bis Rechenzentrum

Intel / AMD (CPU)



Machine Learning (ML)
Artificial Intelligence (AI)

GPU-Accelerated Applications



Container Technologies

Supercomputer

GPU