Präsentation 1:
Linux Namespaces

Martin Leischner
4. Oktober 2021

Namespaces + cgroups

Namespaces: Leichtgewichtige Prozess-Virtualisierungen.

- **Verschiedene Prozesse sollen verschiedene Sichten auf die Systemressourcen haben.**
- **Hierdurch lassen sich verschiedene Prozesse auf einem System voneinander isolieren. (Prozess-Virtualisierung)**
- **Namespaces sind ab Kernel-Version 2.4.19 (Jahr 2002) direkt in Linux eingebaut. Daher ist kein Hypervisor für die Isolation von Prozessen notwendig.**

cgroups: "control groups" ist eine Ressourcen-Management-Lösung.

- **Einer Gruppe von Prozessen werden Ressourcen zugewiesen. Die Nutzung der Ressourcen wird überwacht. Insbesondere kann sie limitiert und priorisiert werden.**
- **Erste Ansätze ab Linux Kernel-Version 2.6.24. Ab Ubuntu 12.10.**

Anwendungen: Linux Containers (LXC), Docker.

Wichtige Namespaces

- **PID Namespace**

  Gruppiert Prozesse, die sich gegenseitig sehen.

  Keine Sicht nach außen.

  Erster Prozess im jeweiligen Namespace bekommt die **PID 1**.

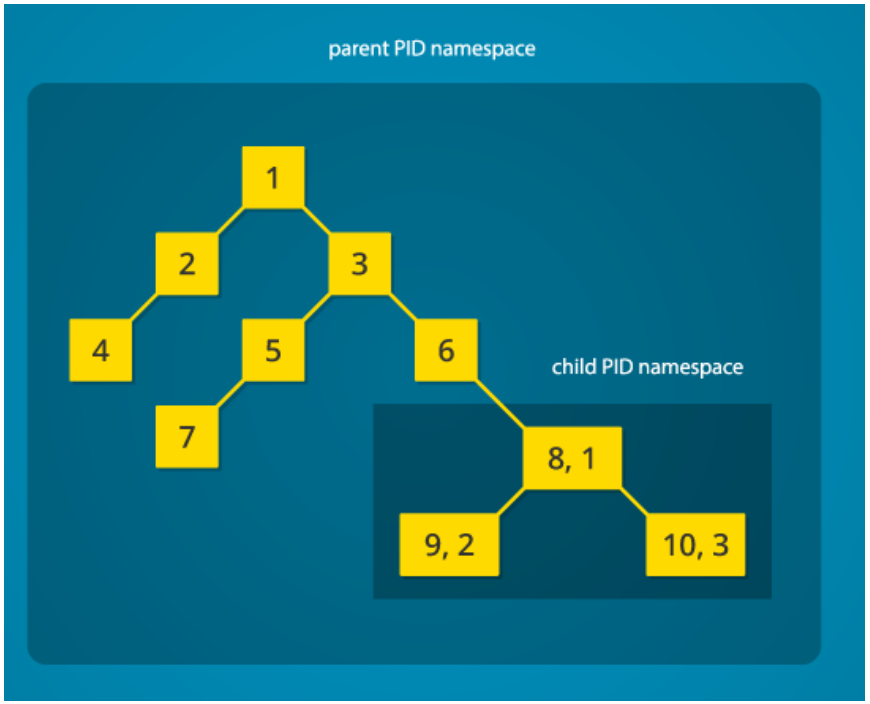


Abb.: Mahmud Ridwan, https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces

- **Network Namespace**

  Die Interfaces im Namespace können im Namespace uneingeschränkt genutzt werden, ohne in Konflikt mit den Interfaces außerhalb des Namespaces zu geraten.

- **Mount Namespace**

  Filesysteme können gemounted werden, ohne dass das Hostsystem davon betroffen ist.

- **… + Cgroup / IPC / User / UTS Namespace**

Kurzdemo

Erzeugung eines neuen PID-Namespaces und Wechsel in diesen.

Hierzu wird der Linux-Befehl unshare verwendet (http://manpages.ubuntu.com/manpages/xenial/man1/unshare.1.html)

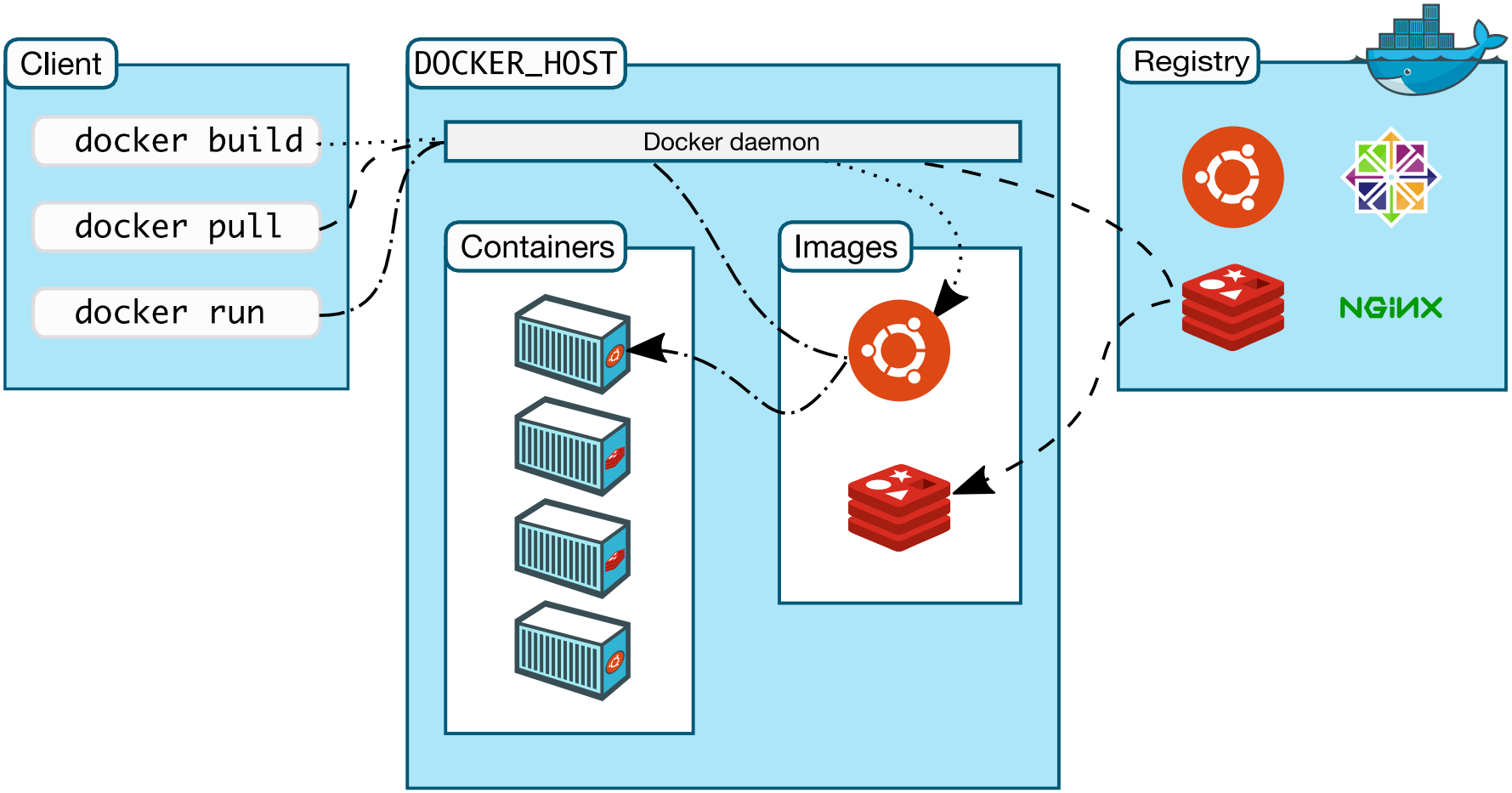
unshare

--fork	Das angegebene Programm als Kind abspalten.
--pid	Den PID Namespace abspalten. Anmerkung: Mit `--net` wird der Network Namespace abgespalten.
--mount-proc	Das proc Filesystem (= Schnittstelle zum Kernel) an den Mountpoint `/proc` anmounten
program	Programm, das als erstes im neuen PID-Namespace gestartet wird.

Dockerarchitektur

Martin Leischner
4. Oktober 2021

Docker Architektur



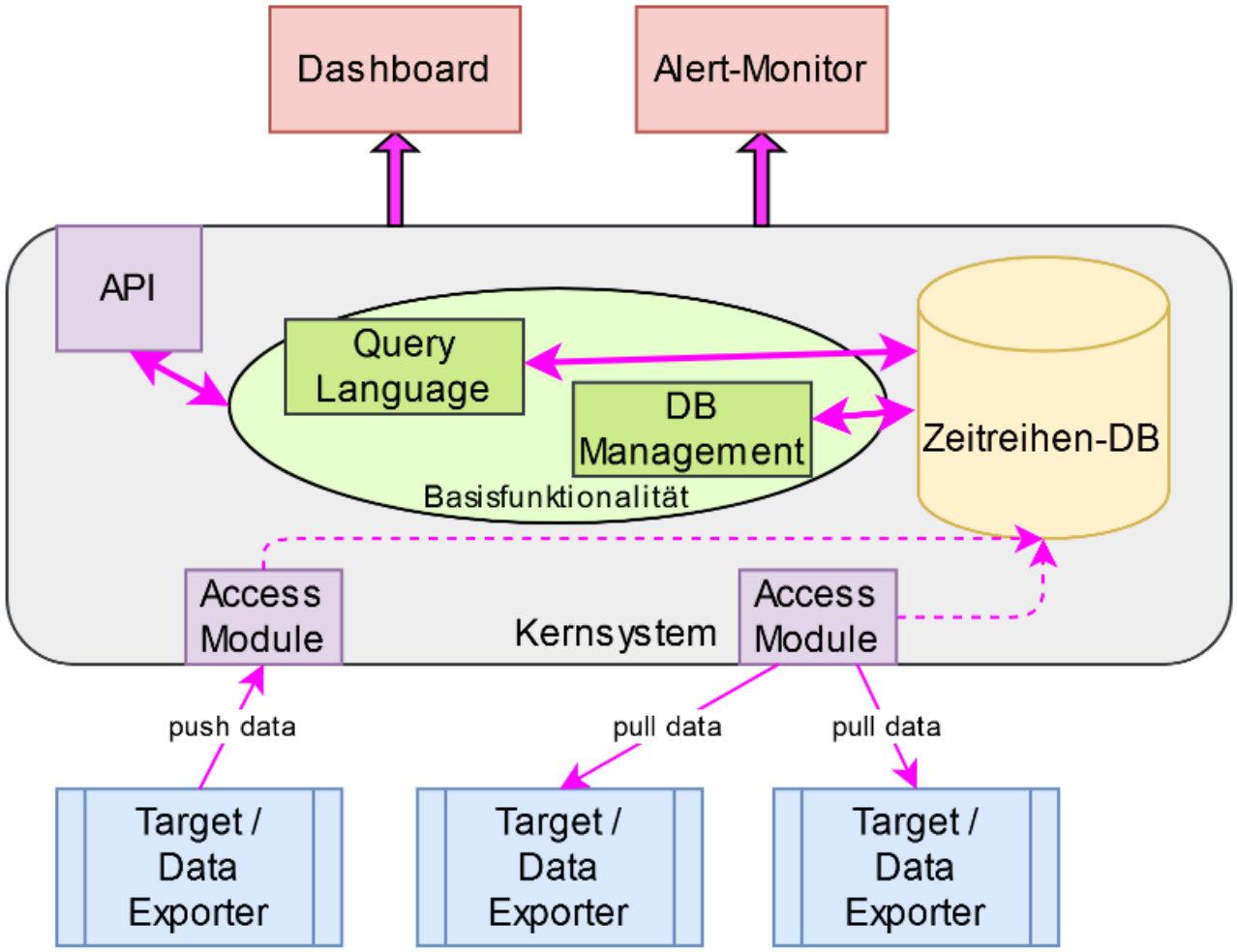
from: docker docs, https://docs.docker.com/get-started/overview/

Wichtige Dockerkommandos

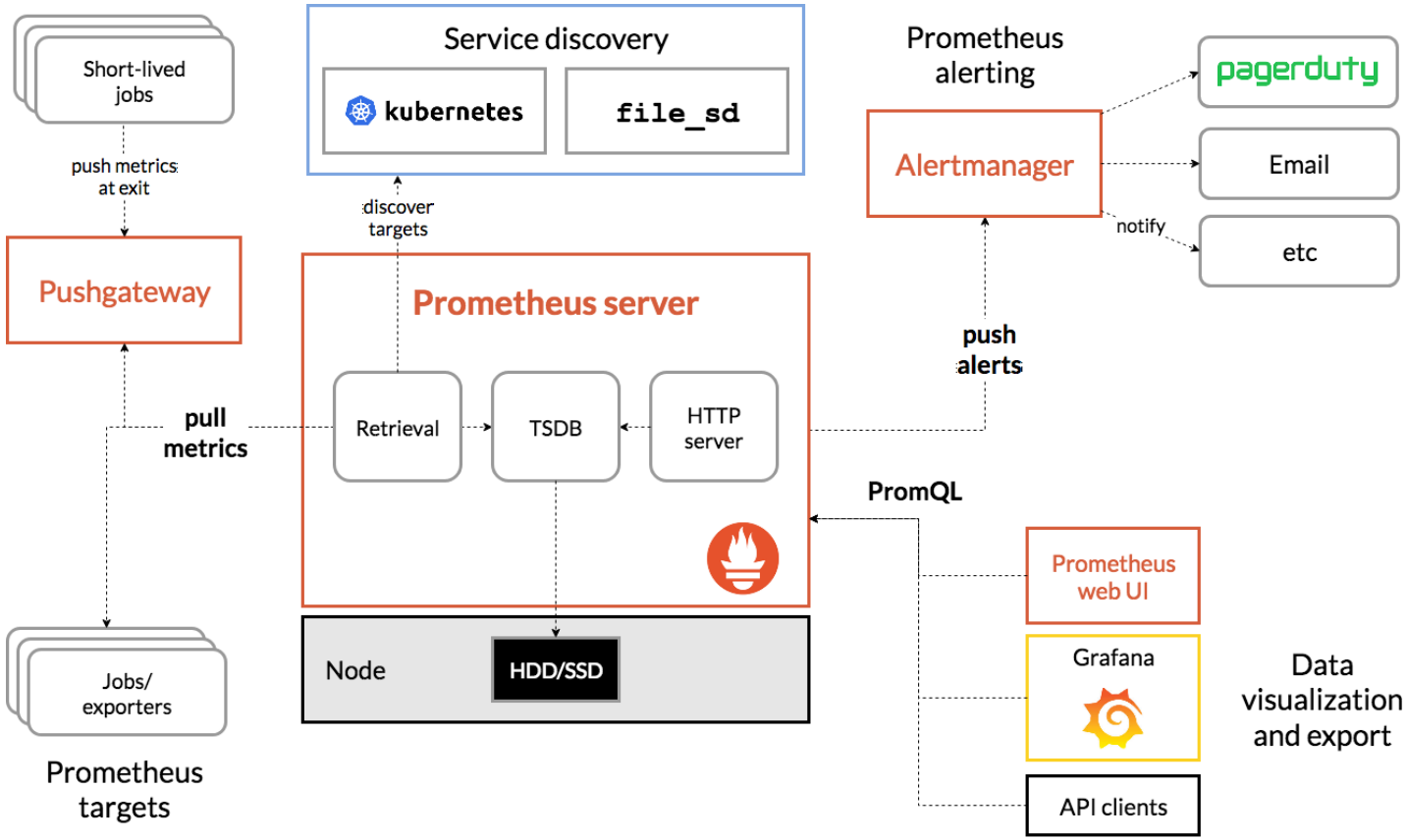
`docker version`	Version von Docker
`docker ps`	Laufende Container listen
`docker run -it ubuntu`	Image Ubuntu interaktiv als Container laufen lassen. Image wird automatisch ge`pull`ed
`docker ps -a`	Alle Container (auch nicht laufende) listen
`docker images`	Alle Images listen
`docker rmi ubuntu`	Image `ubuntu` löschen
`docker rm e2a2412574cb`	Container `e2a2412574cb` löschen. Vorher stoppen.
`docker start 98838dfffd93`	Container `98838dfffd93` starten
`docker exec -it jolly_hawking /bin/bash`	In den Container `jolly_hawking` gehen und die bash ausführen
`docker tag ubuntu:latest ubuntu:v2.3`	Dem Image einen neuen Tag hinzufügen

Architektur von Monitoringsystemen

Martin Leischner
4. Oktober 2021

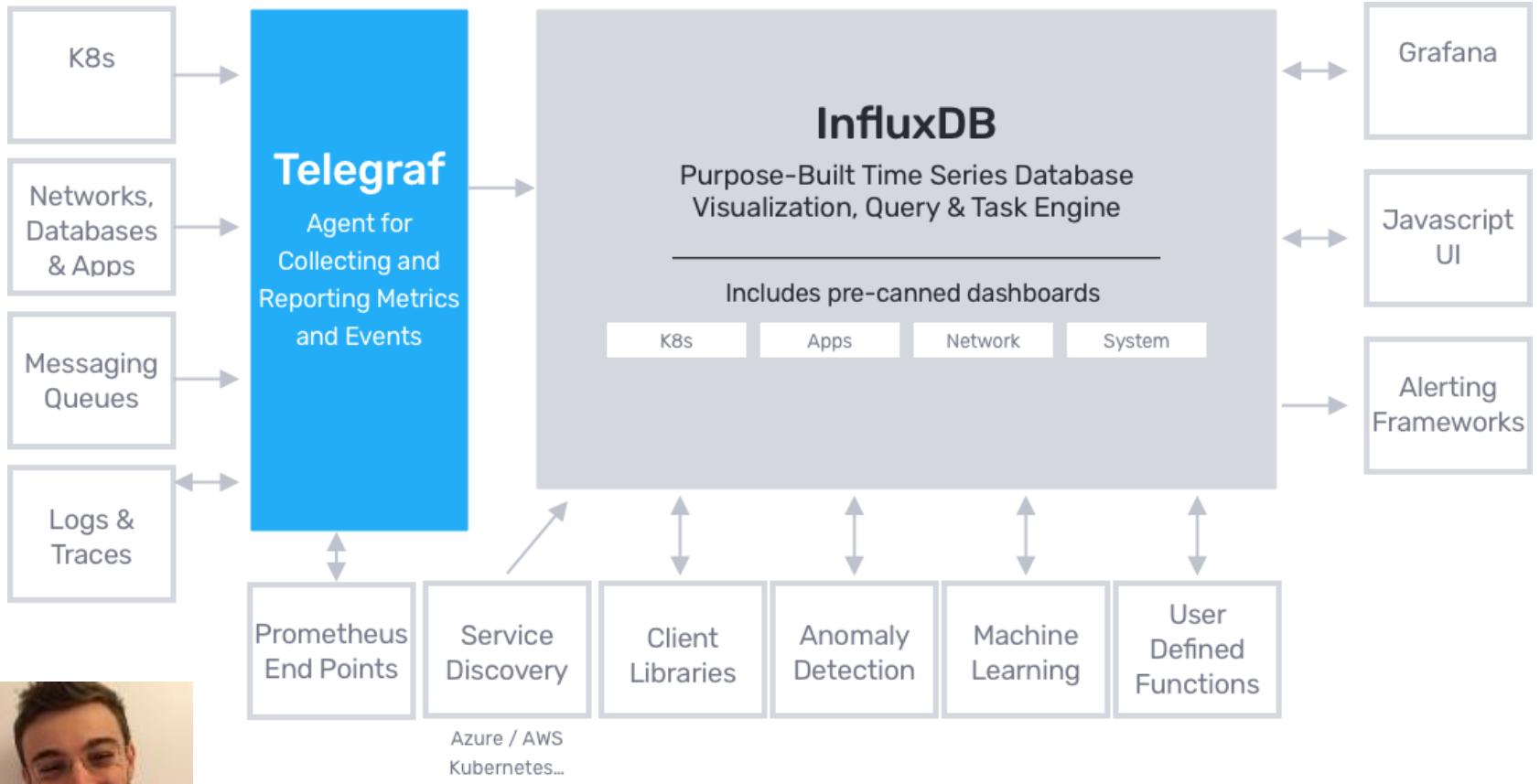

Dashboard
Alert-Monitor
API
Query Language
DB Management
Basisfunktionalität
Zeitreihen-DB
Access Module
Kernsystem
Access Module
push data
pull data
pull data
Target / Data Exporter
Target / Data Exporter
Target / Data Exporter

Prometheus Architektur



from: Prometheus, https://prometheus.io/docs/introduction/overview/

InfluxDB 2.0 Architektur



from: Schkn (Antoine Solnichkin), InfluxDays London 2019 Recap, https://devconnected.com/influxdays-london-2019-recap/, 2019