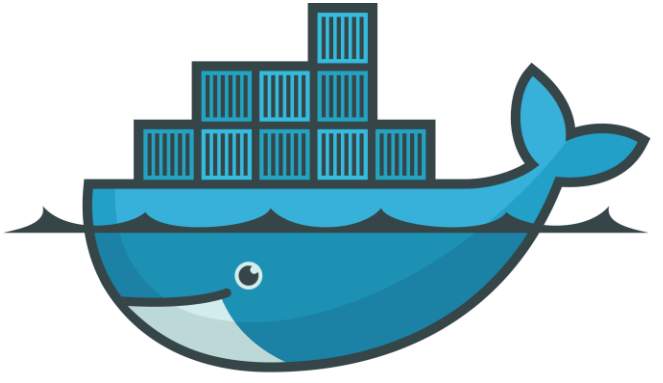


Docker

SNEK 2018 | Bernhard Mayr

Bernhard Mayr

mailto:// bernhard@mayr.io
https:// www.softaware.at
twitter:// @bemayr
github:// @bemayr

softaware

Aware of your ideas.
Developing your software.

- studying **psychology**
  @ university of innsbruck
- studied **software engineering**
  @ fh hagenberg
- **working**
  @ softaware gmbh

#types #docker #typescript #ux #angular #windows #haskell #continousintegration #dotnetcore #types #graphs #dx #idris #xplatform #azure #purescript #droneio #statecharts #rx #continuousdeployment

Begriffsdefinitionen
(Henne-Ei-Problem)

Virtualisierung

„Erstellens einer simulierten Computingumgebung statt einer physischen Version”

Image

„Bauanleitung für das Erstellen eines Containers”

Container

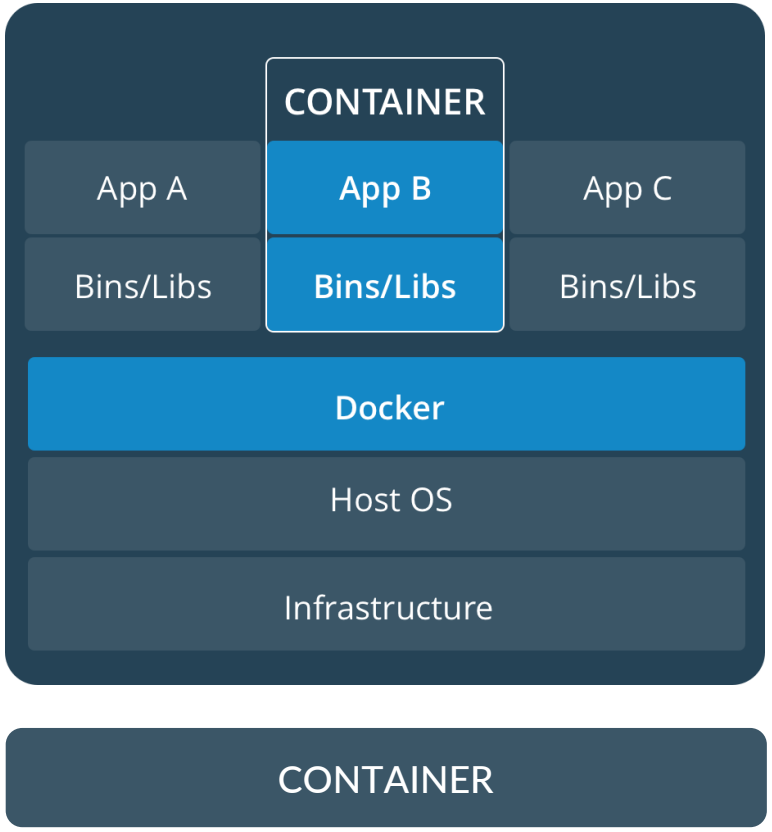
„Ein Container ist eine lauffähige Instanz eines Images.”

Registry

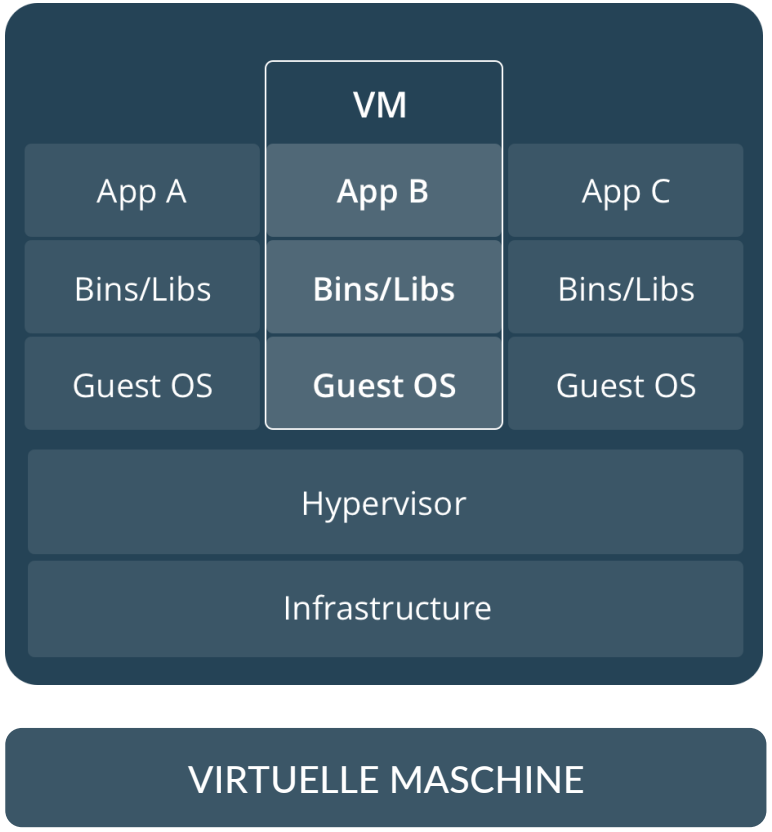
„Verzeichnis für fertige Docker Images”

Virtualisierung

Container- vs. Vollvirtualisierung



CONTAINER



VIRTUELLE MASCHINE

Bildquelle: https://www.docker.com/what-container

Ein paar Worte Geschichte zur Virtualisierung

- Virtualisierung seit den 1960ern auf IBMs Mainframes
- 2001: erste Virtualisierung ohne Vollvirtualisierung
- 2006: Aufnahme von `cgroups` in den Linux-Kernel
- 2008: Implementierung von `namespaces`
- 2008: Entwicklung von *LXC*
- 2013: Docker wird vorgestellt
- 2014: Docker 1.0 mit *libcontainer* statt *LXC*
- 2015: Docker tritt der *Open Container Initiative* bei

Motivation für den Einsatz von Containern

- Verpacken von Anwendungen in systemunabhängige Pakete
- Portabilität (~~“works on my machine”~~)
- Wiederverwendbarkeit
- Vereinfachung von CD/CI-Prozessen
- Leichtgewichtigkeit
  - Speicherverbrauch
  - Startzeit
  - Prozessorauslastung (Stromverbrauch in großen Datenzentren)

Docker, Inc.

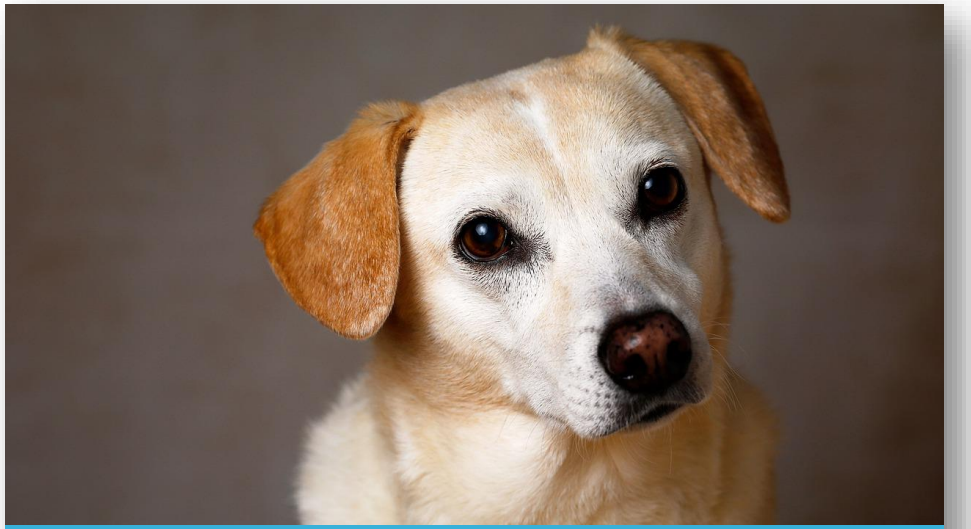
- Docker Community Edition
- Docker Enterprise Edition
- Docker vs. Moby
- Open Container Initiative (OCI)
  - https://www.opencontainers.org/

Was sind Container?

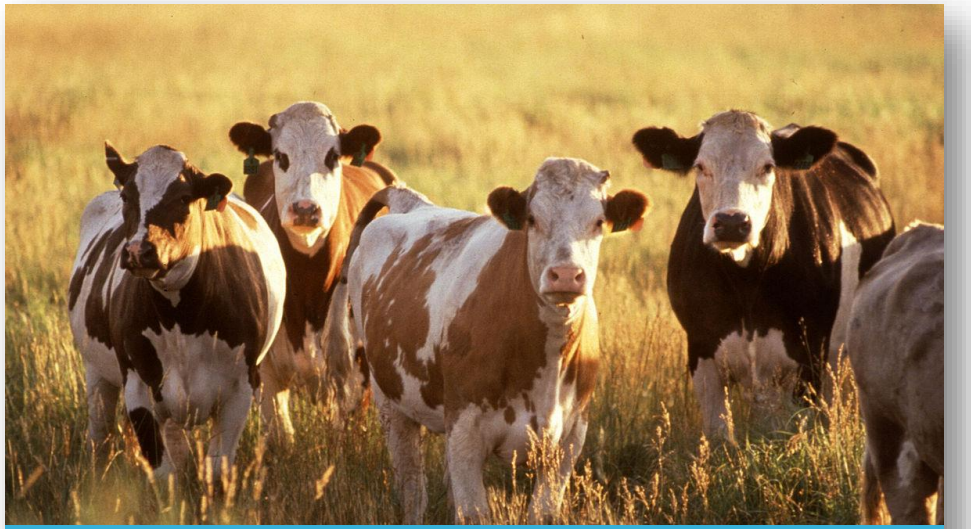
“a standardized unit of software”

- Container sind **zustandslos**
- Kommunikation mit der Außenwelt erfolgt über
  - Port-Mappings
  - Speicher-Mounts
- Konfiguration der Container erfolgt idealerweise über Umgebungsvariablen
- *Pet vs. Cattle* Analogie

Pet vs. Cattle Analogie

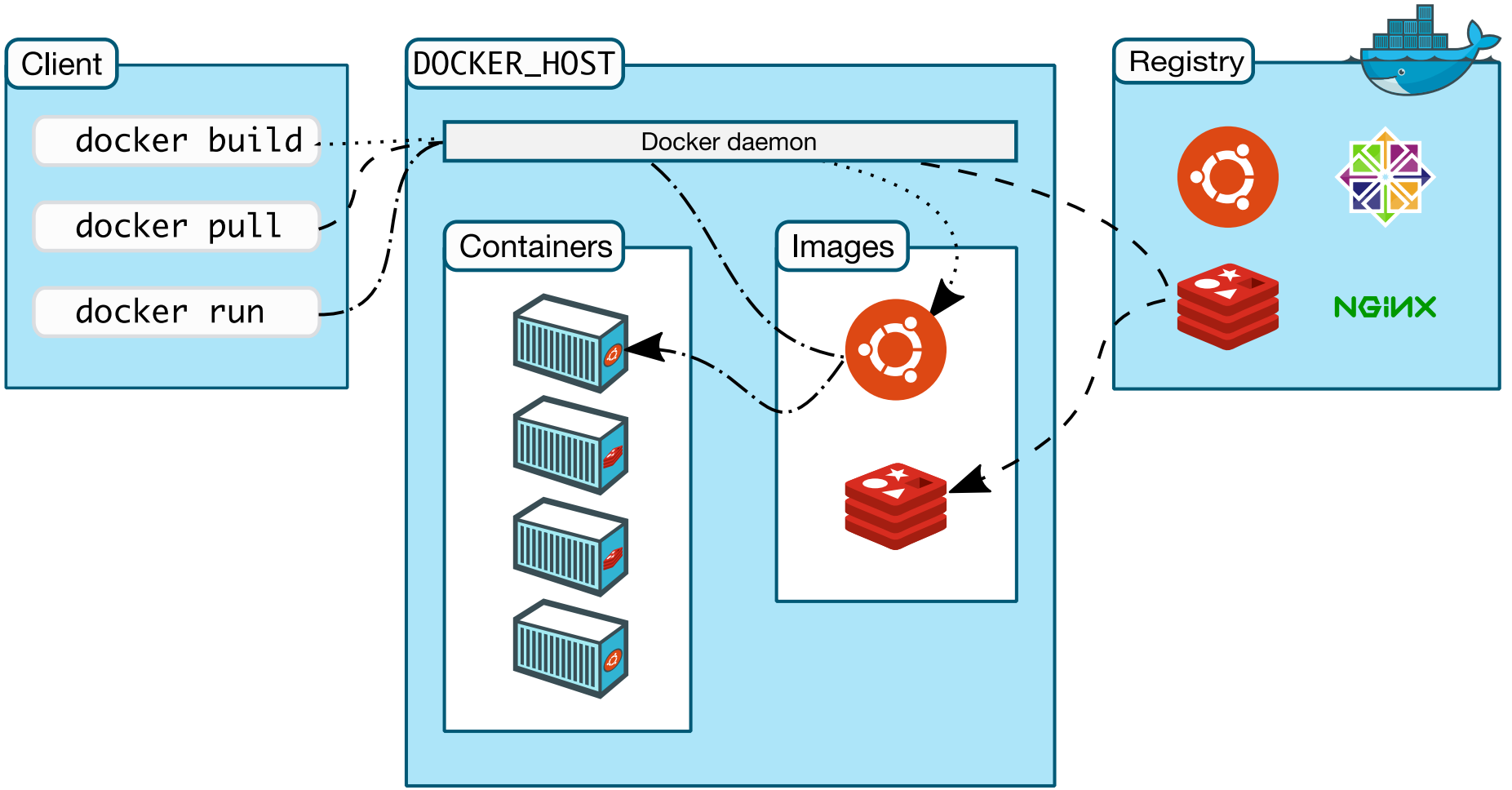


PFLEGEN



ERSETZEN

Docker Komponenten



Bildquelle: https://docs.docker.com/engine/docker-overview/#docker-architecture

Docker Registry

- Verzeichnis in dem fertige Images liegen, die mit `docker pull` oder `docker run` gestartet werden
- mit `docker push` können dort auch eigene Images geteilt werden

- Docker Hub
- Docker Cloud
- Docker Store

DEMO Docker Registry

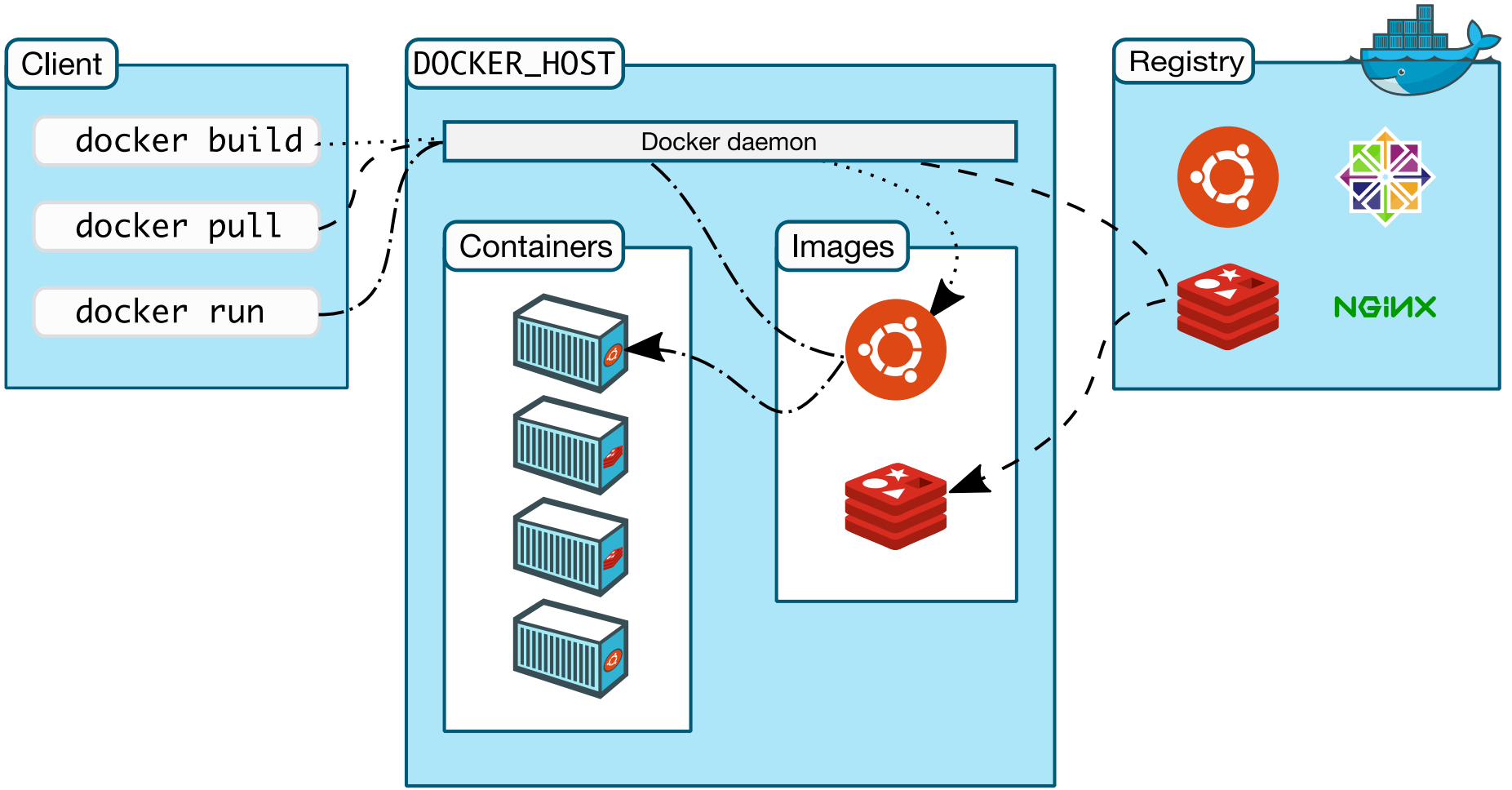
bemayr/talk.intro-to-docker

Image vs. Container

- Images sind die “Bauanleitungen” für Docker um Container zu erstellen
- *„Ein Container ist eine lauffähige Instanz eines Images.“*

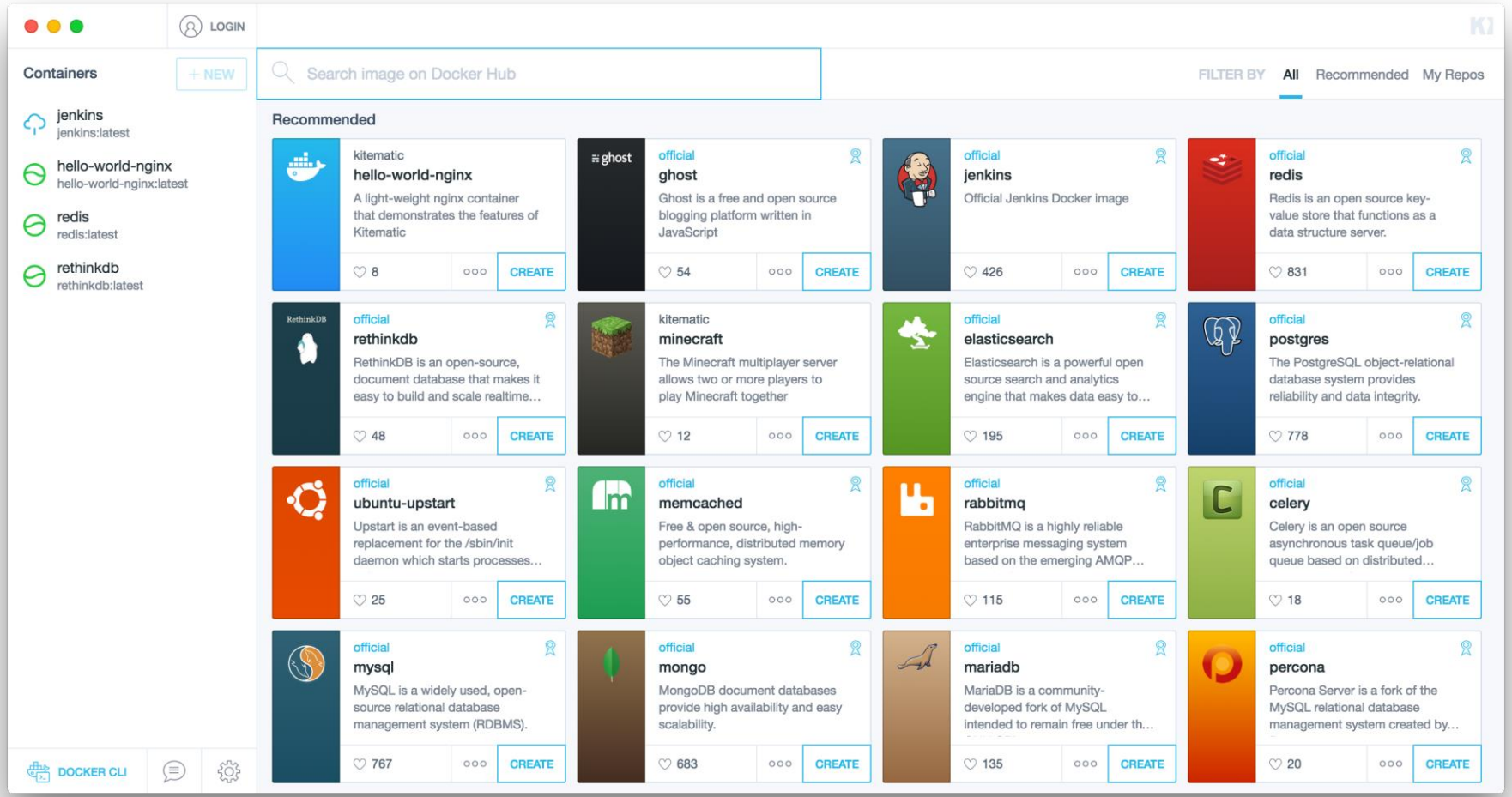
- Images
  - bauen aufeinander auf
  - werden idealerweise mithilfe von **Dockerfiles** erstellt
  - für die meisten Fälle gibt es bereits **fertige Images** im Docker Hub
  - **Tags** werden für Versionen/Varianten verwendet

Docker Client



Bildquelle: https://docs.docker.com/engine/docker-overview/#docker-architecture

Docker Kitematic



Bildquelle: https://github.com/docker/kitematic

Docker CLI

```
CLI + PowerPoint => :/
      PowerShell => :)
```

DEMO Docker CLI

bemayr/talk.intro-to-docker

Erstellen von Docker Images

- wenn bestehende Images nicht reichen, dann können Images selbst gebaut werden
- bestehende Images als Basis verwenden
- zwei Möglichkeiten
  - `docker commit […]`
  - Dockerfiles + `docker build […]`
- empfohlener Weg für “Nicht-Linux-Profis”
  - `docker run --rm -it […]`
  - Kommandos ausführen und ausprobieren
  - nebenbei ein *Dockerfile* dazu erstellen

DEMO Erstellen eines Images

bemayr/talk.intro-to-docker

DEMO Dockerfiles

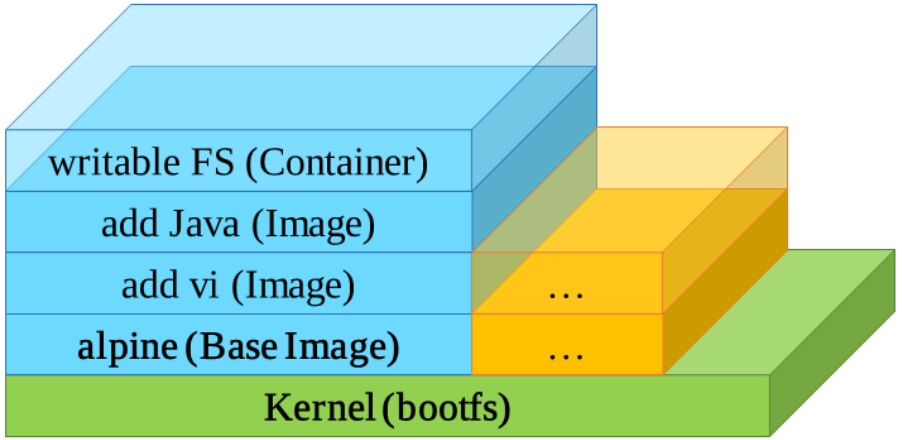
bemayr/talk.intro-to-docker

Dockerfiles: Best Practices

- Build-Kontext bedenken (.dockerignore)
- Installation von nicht benötigten Paketen vermeiden
- jeder Container soll **eine Aufgabe** haben!
- Anzahl der Schichten minimieren (`RUN`, `COPY`, `ADD`)
- Kommandozeilenargumente sortieren
- Ausnützen des Build Caches
- Multi-Stage Builds
- der ideale Container ist “flüchtig”

Docker Dateisystem

- Union File System
- schichtenbasiert
- jede Schicht bekommt eine kombinierte Sicht auf alle darunterliegenden
- Wiederverwendung von identen Schichten (Speichereffizienz)



Volumes vs. *Bind Mounts*

DEMO Docker for Windows

bemayr/talk.intro-to-docker

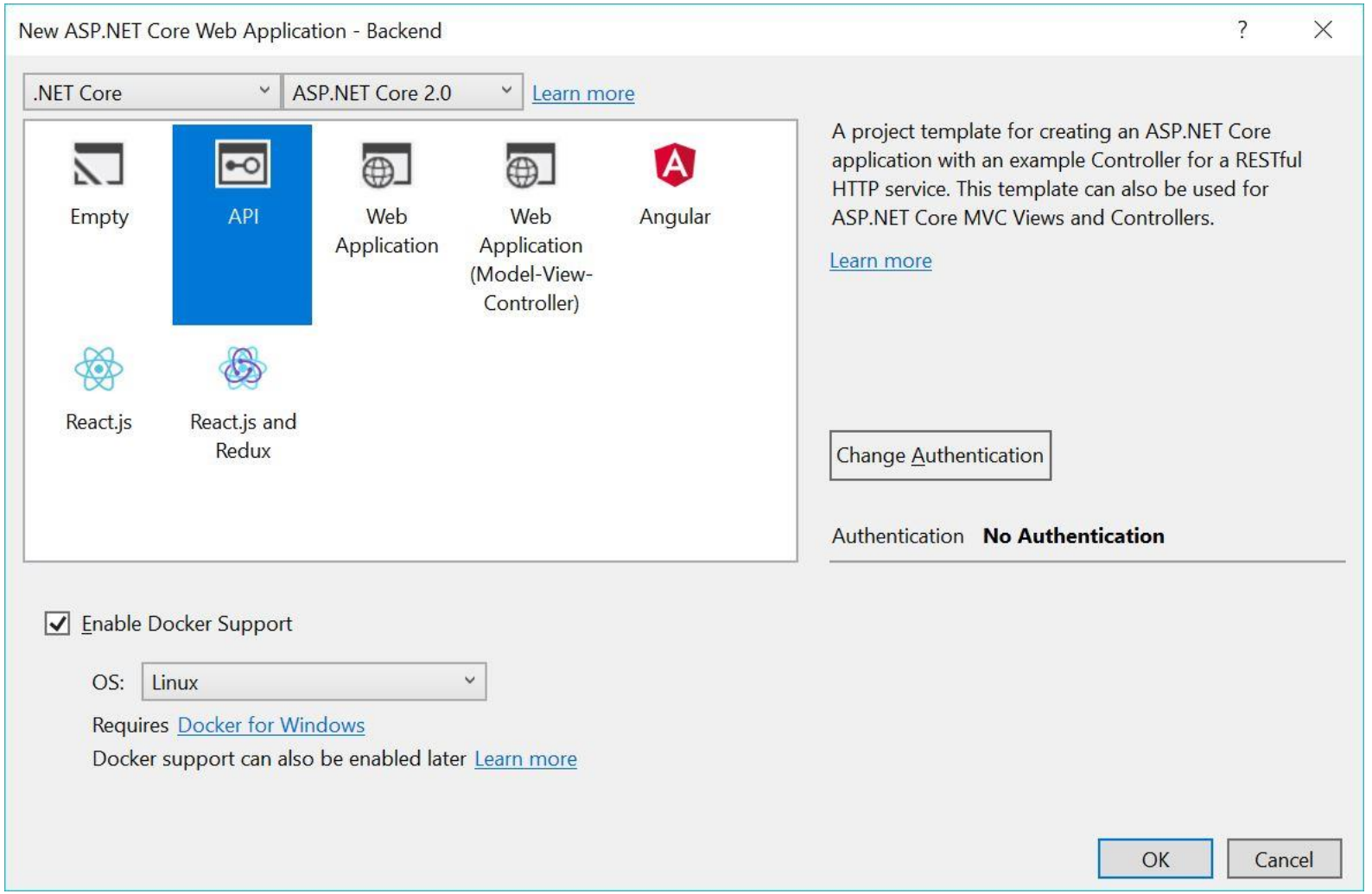
Windows und Docker

- Docker for Windows
  - linuxkit-VM in Hyper-V
  - Client auf Windows
- Docker Toolbox
- Windows Container
  - Hyper-V Container
  - Windows Server Container

docker-compose

- Die meisten Anwendungen bestehen nicht aus einem Container, sondern z.B. Backend, Frontend und Datenbank.
- Docker-Philosophie: Anwendung auf mehrere Container (Services) aufteilen
- *docker-compose* ist ein Werkzeug um diese Services gemeinsam zu verwalten
- Definition der Services mithilfe von YAML
- Scaling

docker-compose (VS 2017)



DEMO docker-compose

bemayr/talk.intro-to-docker

Datenbanken im Container

- Datenbanken in Docker haben einen besonderen Status
- da Container flüchtig/zustandslos sind, können sie per se keine Daten speichern
- Lösung des Problems: *Docker Volumes* und den DB-Container als reines Service sehen

```
docker volume create --name db_data
docker run
    -d
    -v db_data:/container/path/for/volume
    <container-image> <my-startup-command>
```

SQL Server im Container

- verfügbar als Linux-Container: https://hub.docker.com/r/microsoft/mssql-server-linux/
- Achtung: mindestens 2GB RAM zuweisen!

```
docker run
   -d
   -e 'ACCEPT_EULA=Y‘
   -e 'SA_PASSWORD=<password>‘
   -p 1433:1433
   microsoft/mssql-server-linux
```

DEMO microsoft/mssql-server-linux

DEMO posh-docker (Autocomplete)

Docker aufräumen

- einfaches Bereinigen von belegtem Speicherplatz ist seit Docker Version 1.13 möglich
  - gestoppte Container
  - nicht verwendete Images
  - übriggebliebene Speicher-Volumes

```
docker system df            # Show docker disk usage
docker container
       volume
       image
       system    prune    # Clean Resource
```

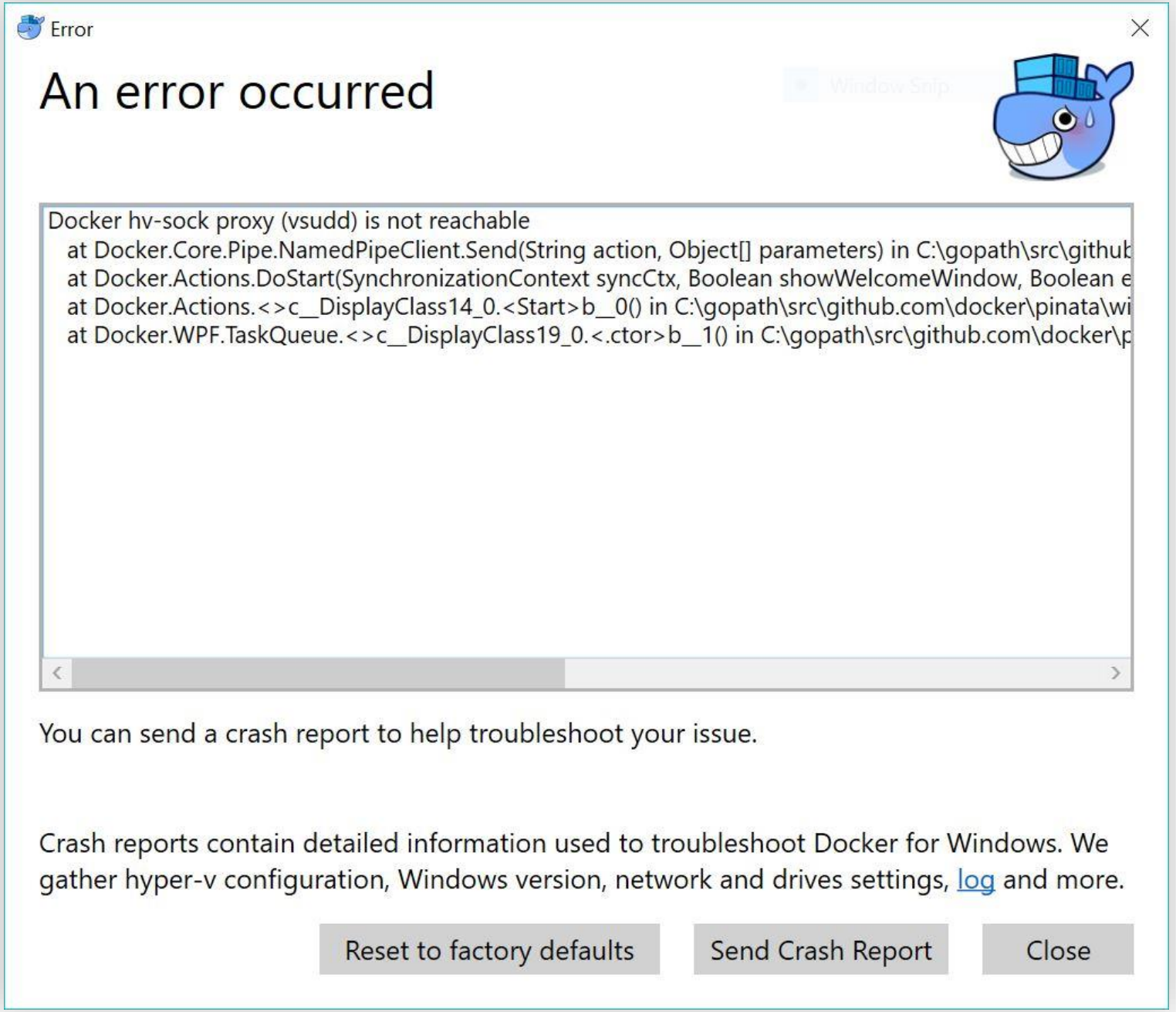
Weitere Anwendungsmöglichkeiten
für Softwareentwickler

- Softwareevaluierung
- Plattformunabhängige CLI-Anwendungen
- Containerbasierte Integrationstests
- Plattformübergreifende Übersetzung
- IDE in a Container
- Deterministische Build-Systeme
  - https://github.com/softawaregmbh/docker-webdev

- Produktivsysteme (Swarm, Kubernetes, …)

Persönliche Tipps/Infos

- Aufpassen bei Docker Updates
  - ist mittlerweile allerdings wesentlich besser


Error
An error occurred
Docker hv-sock proxy (vsudd) is not reachable
at Docker.Core.Pipe.NamedPipeClient.Send(String action, Object[] parameters) in C:\gopath\src\github
at Docker.Actions.DoStart(SynchronizationContext syncCtx, Boolean showWelcomeWindow, Boolean e
at Docker.Actions.<>c__DisplayClass14_0.<Start>b__0() in C:\gopath\src\github.com\docker\pinata\wi
at Docker.WPF.TaskQueue.<>c__DisplayClass19_0.<.ctor>b__1() in C:\gopath\src\github.com\docker\p
You can send a crash report to help troubleshoot your issue.
Crash reports contain detailed information used to troubleshoot Docker for Windows. We gather hyper-v configuration, Windows version, network and drives settings, log and more.
Reset to factory defaults
Send Crash Report
Close

Persönliche Tipps/Infos

- Aufpassen bei Docker Updates
  - ist mittlerweile allerdings wesentlich besser
- *Shared Drives* nicht vergessen freizugeben
- Aufpassen bei Netzwerkinterfaces im Container (`0.0.0.0`)
- File-Watches funktionieren (noch) nicht
- Container eignen sich leider nicht/nur schwer um zeitzonenabhängiges Verhalten zu testen
- saubere Benennung mit Tags ist sehr wichtig
- Benutzer in Containern

DEMO Was mache ich mit Docker?

bemayr/talk.intro-to-docker

mailto:// bernhard@mayr.io
https:// www.softaware.at
twitter:// @bemayr
github:// @bemayr

soft aware

Aware of your ideas.
Developing your software.