Docker 1. Schritte

nicht nur für Connections und Domino Admins

Christoph Stoettner
+49 173 8588719
christoph.stoettner@panagenda.com

Christoph Stoettner

+49 173 8588719
christoph.stoettner@panagenda.com
linkedin.com/in/christophstoettner
stoeps.de
christophstoettner
@stoeps

IBM CHAMPION

- Senior Consultant bei panagenda
  - IBM Domino seit 1999
  - IBM Connections seit 2009
- Schwerpunkt
  - Migrationen, Installationen
  - Performance Analysen
- Focus auf
  - Monitoring, Security
  - panagenda ConnectionsExpert

Wer kennt das?

"In Produktion funktioniert ABC nicht!"

"… funktioniert super auf meiner Maschine!"

"Dann ist es jetzt ein Problem der **Admins**!"


WORKS ON MY MACHINE
38
makeameme.org

Verschiedene Technologien

- Jeder Admin pflegt eine große Sammlung

- Bare Metal
- On Premises
-  Linux
- Traditionelle Applikationen

- Virtualisierung
- ☁ Cloud
- Windows
- Microservices

… und verschiedene Abteilungen

- Diese haben verschiedenste Anforderungen

Entwickler

Freiheit, Anwendungen schnell zu erstellen und zu implementieren

Definieren und Verpacken von Anwendungsanforderungen

IT Betrieb

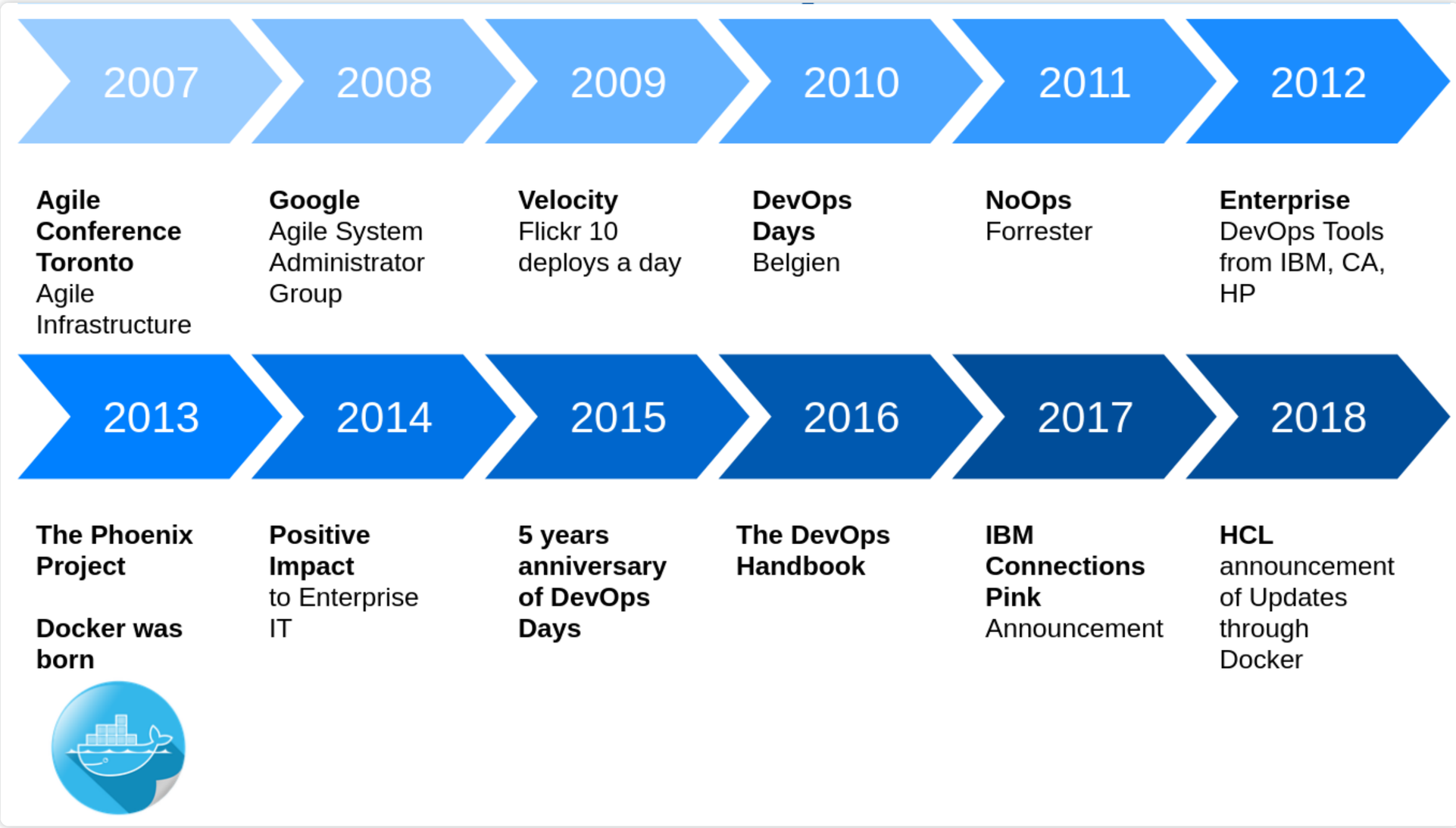
schnell und flexibel auf sich ändernde Bedürfnisse reagieren

Standardisierung, Absicherung und Verwaltung

Mythos

	Microservices	Traditionelle Applikationen
Cloud oder Neue Infrastruktur	Sie sind entweder hier …	
Alte Infrastruktur		… oder hier.

2009 DevOps Movement



Agile Manifesto

- **Individuen und Interaktionen**
  - *mehr als Prozesse und Werkzeuge*
- **Funktionierende Software**
  - *mehr als umfassende Dokumentation*
- **Zusammenarbeit mit den Kunden**
  - *mehr als Vertragsverhandlungen*
- **Reagieren auf Veränderungen**
  - *mehr als das Befolgen eines Plans*

Infrastructure as Code

- Schnell auf Änderungen in Systemumgebung reagieren
- Systemadministration mit ähnlichem Vorgehen wie Entwickler
  - Skripte für administrative Aufgaben
  - Konfiguration von Servern in Dateien
  - Ablage in zentralen Versionsverwaltungssystemen
- (Automatisches) Testen von Änderungen
- Monitoring
- Installation wiederholbar

Virtualisierung minimiert Risiken

- Werkzeuge wie Vagrant und Docker können das Risiko von Updates reduzieren
  - keine Abweichungen von Test und Produktion
- Terraform um Vmware oder Cloud Server zu verwalten
- Ansible, Puppet oder Chef
  - Konfiguration
  - Installation

Continuous Delivery durch Automatisierung

panagenda

- Automatisierter Buildprozess
  - Erstellen von Software Updates
  - Aktualisierung von Servern und Betriebssystemen
- Da Software und Konfiguration quasi in produktionsnahen Umgebungen getestet wird
  - Zero Downtime Updates
  - Updates während der normalen Arbeitszeit
  - Keine Nachts- oder Wochenendeinsätze notwendig

CI/CD - Continous Integration

- Voraussetzung
  - Versionsverwaltung mit gemeinsamer Codebasis
  - Automatisierte Übersetzung / Kompilierung
  - Automatisiertes Testen

⊕ Probleme werden laufend entdeckt und können behoben werden

⊕ Frühe Warnung bei Problemen

⊕ Unittests entdecken Fehler meist schon beim Commit

⊕ Ständige Verfügbarkeit von Test-, Demo-, QA-Systemen

⊕ Entwickler checken häufiger ein

⊕ Kleine aber ständige Verbesserungen

CI/CD - Continous Delivery

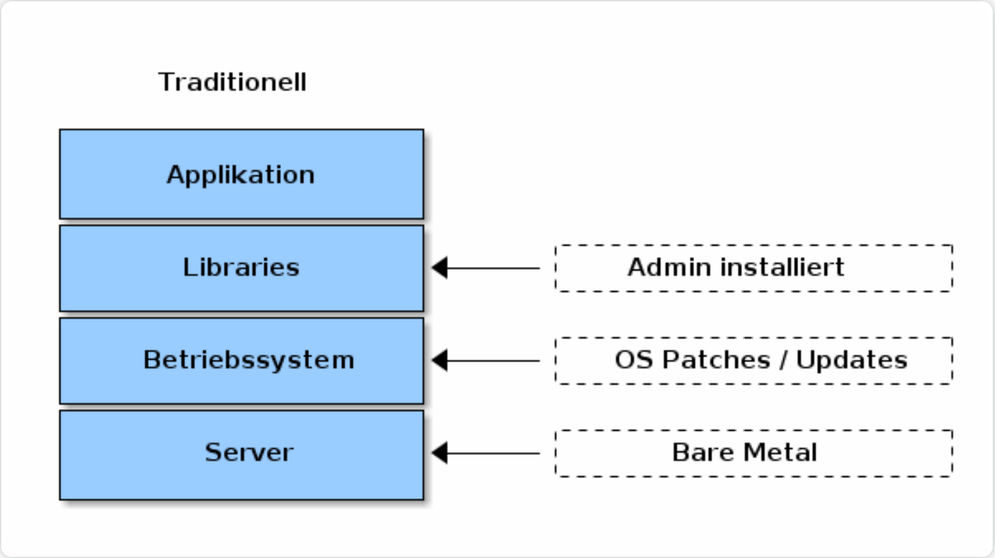
- Automatisierung von Installation und Update
- CI Server wie Jenkins ermöglichen
  - "Nightly", "Test" oder "Release"-Versionen

⊕ Schnelle, zuverlässige und **wiederholbare** Deployments

⊕ Rollback Optionen

Eine Applikation pro physikalischem Server

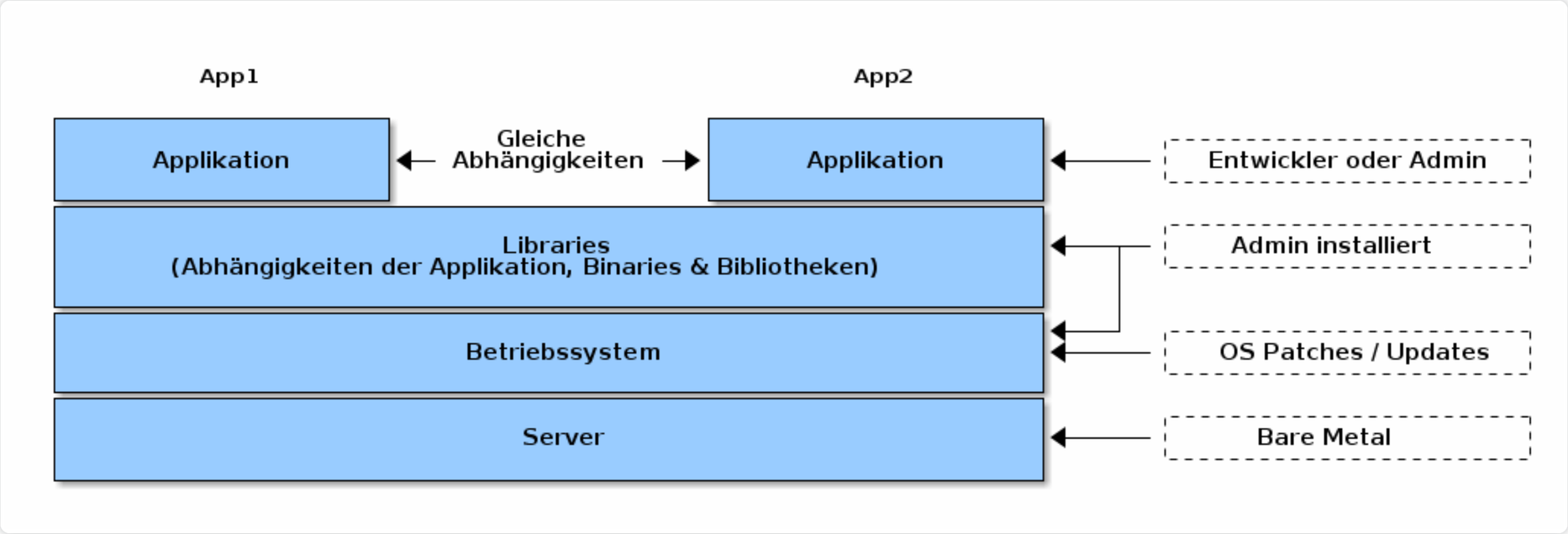
- Langsame Installation
- Hohe Kosten
- Verschwendete Resourcen
- Schwer zu skalieren oder migrieren



Virtualisierung

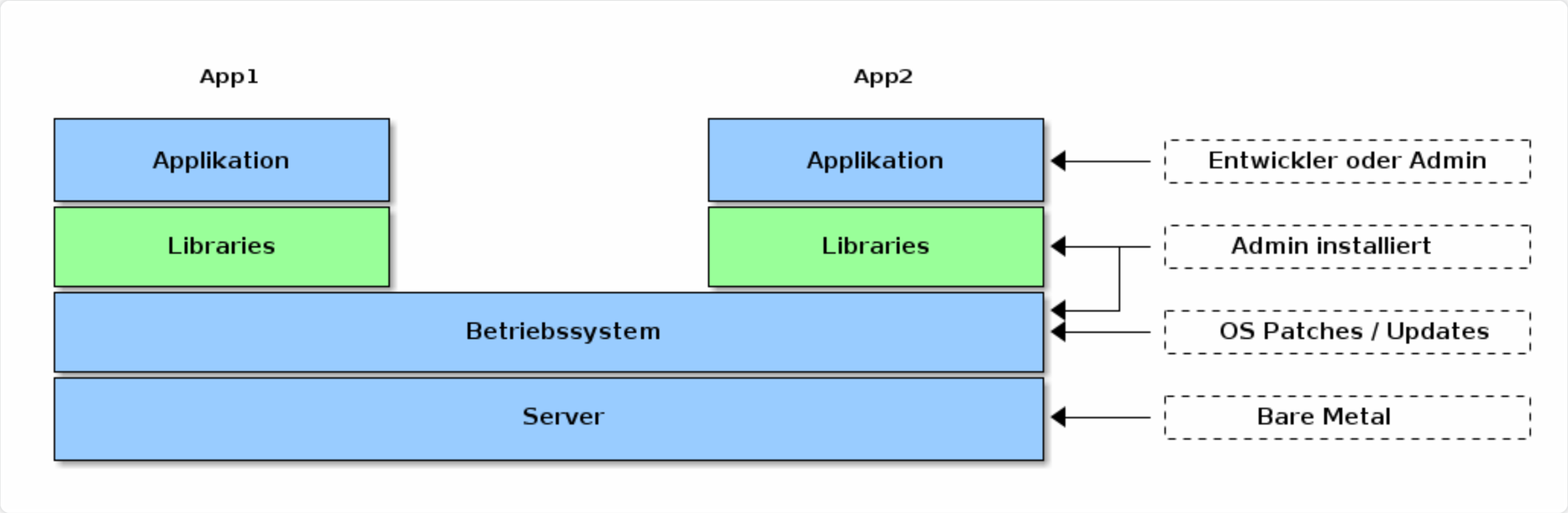
- Emuliert reale Geräte (als Software)
- Hypervisor
  - Erstellen und ausführen von virtuellen Maschinen
  - Virtual Box, VMWare
- Virtuelle Maschine
  - Ein virtueller Computer
- Linux Container
  - Linux OS Virtualisierung
    - Namespaces
    - CGroups

Ressourcenoptimierung



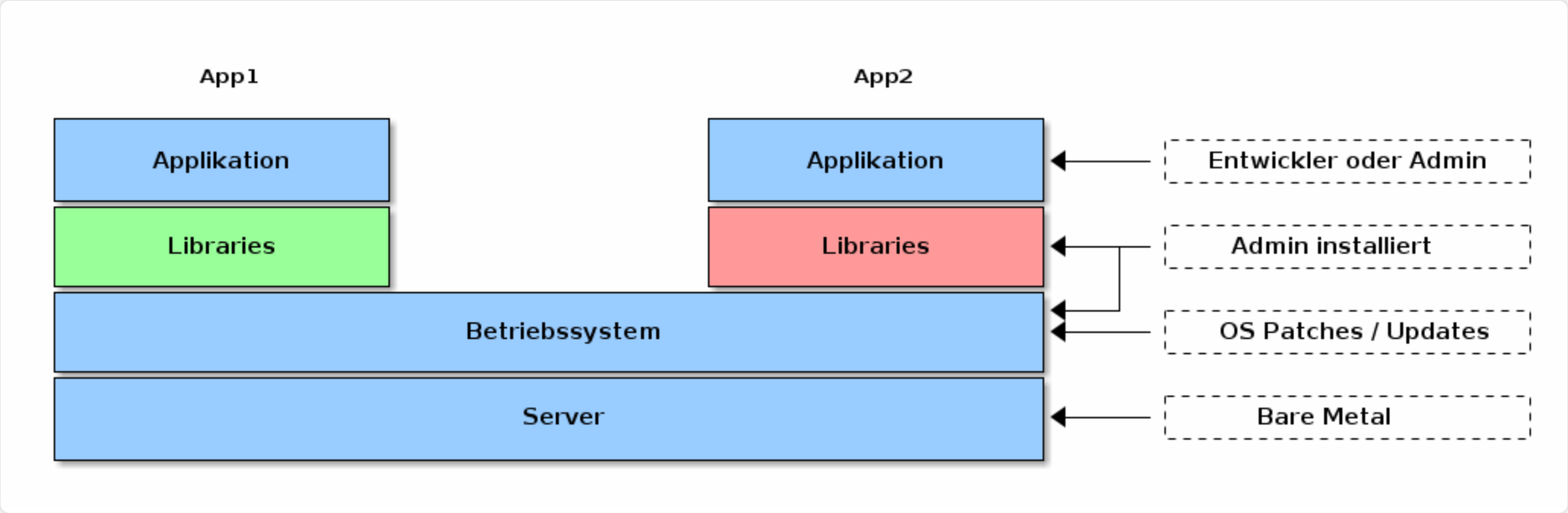
- Applikationen mit gleichen Abhängigkeiten
- Gleiche Applikation, gleiche Version
  - Staging
  - Produktion

Applikationen teilen sich Server



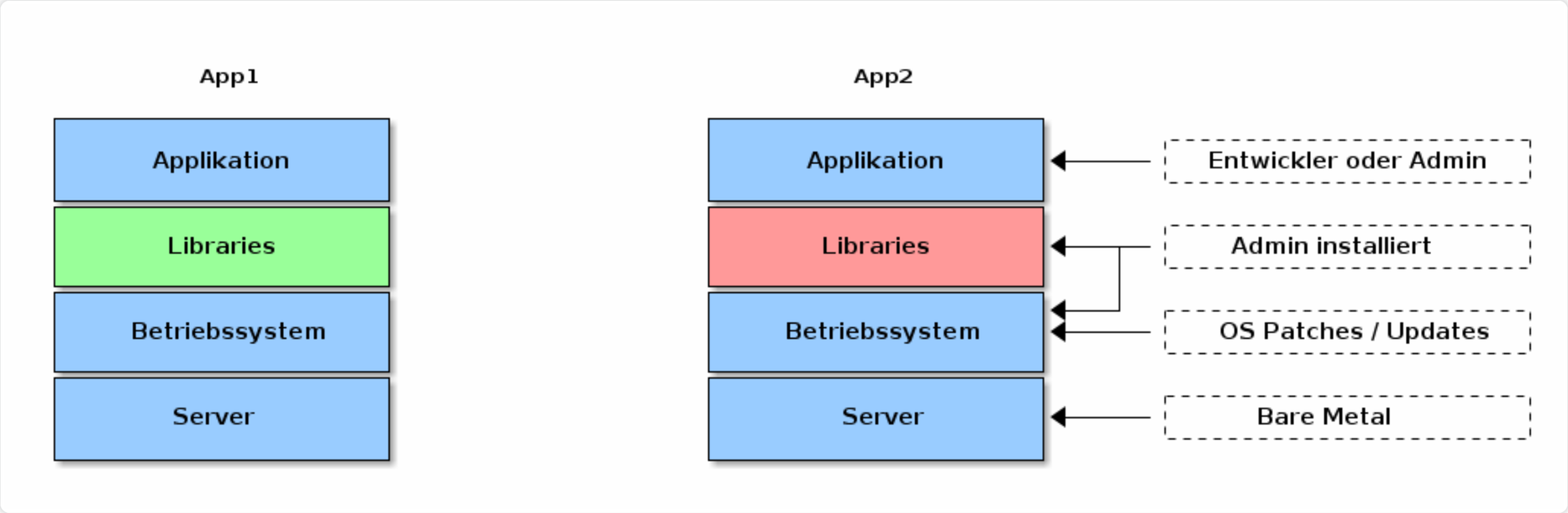
- Unterschiedliche Applikationen oder Versionen mit
  - unterschiedlichen Abhängigkeiten
  - unterschiedliche Versionen von Abhängigkeiten

Unterschiedliche Abhängigkeiten



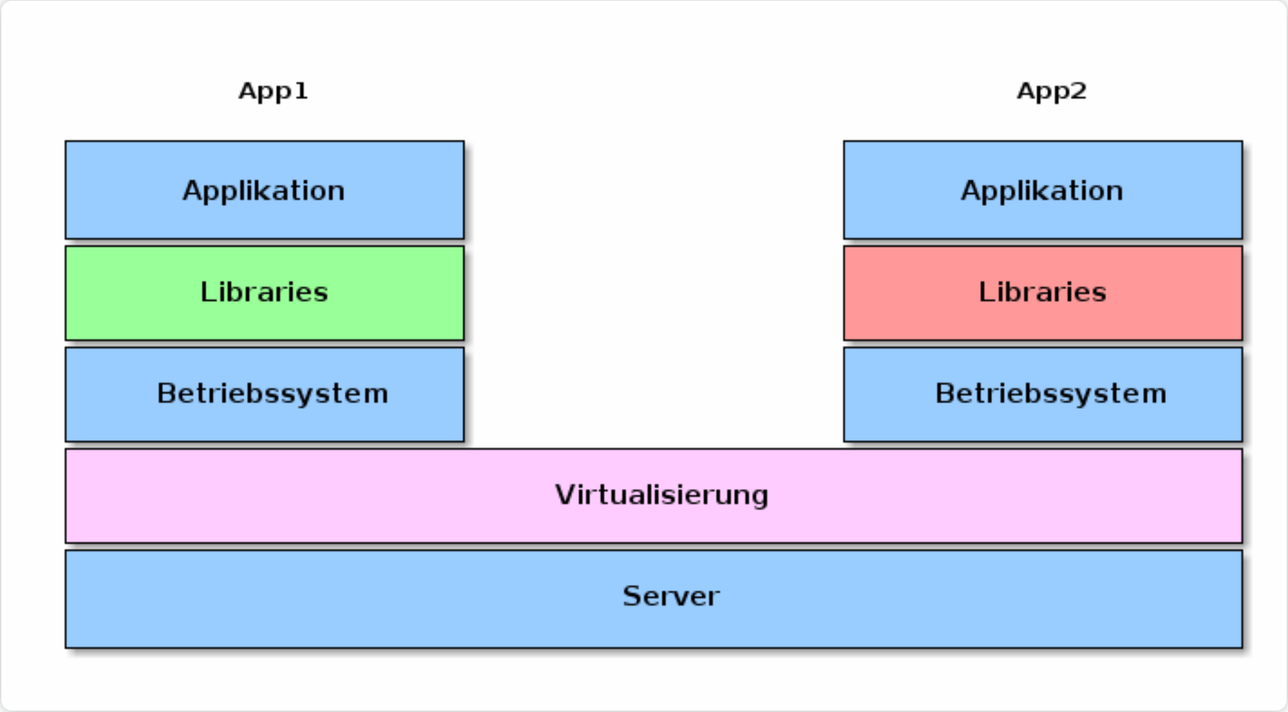
- Abhängigkeiten die sich gegenseitig ausschliessen
- Oft in langen Try&Error Sessions behoben
  - Bis zum nächsten Update

Trennung der Server



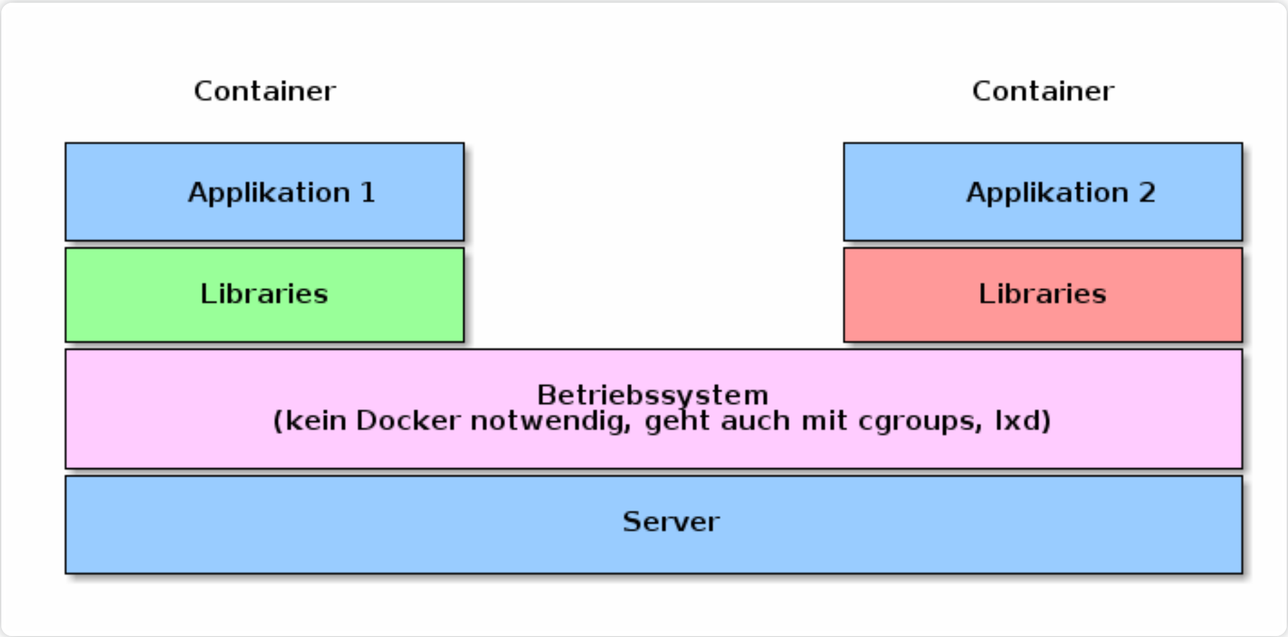
- Großer Overhead / Kostenintensiv
- Lizenzgebühren (OS) beachten
- Puppet / Ansible Kurs buchen

Virtualisierung der Server



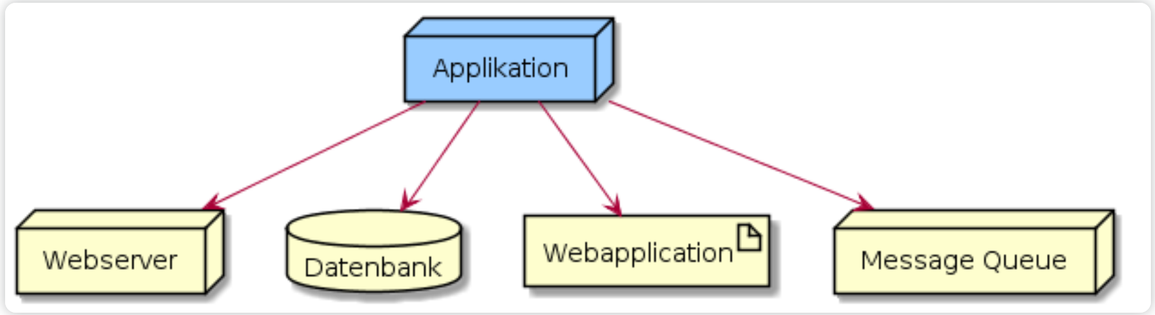
- Betriebssystem Lizenzen
- Automatisierung Patch / Update / Fixes
- Bessere Ausnutzung der Hardware

Container



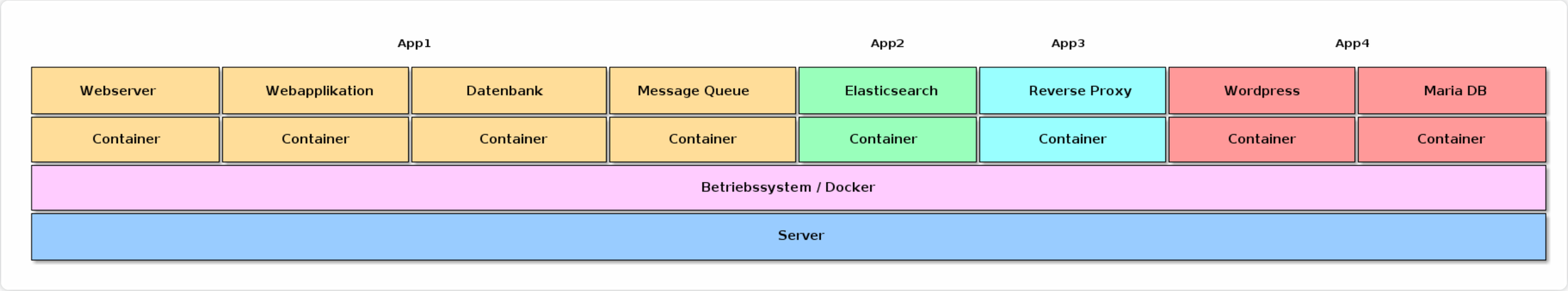
- Immer noch Monolith!

Applikationen



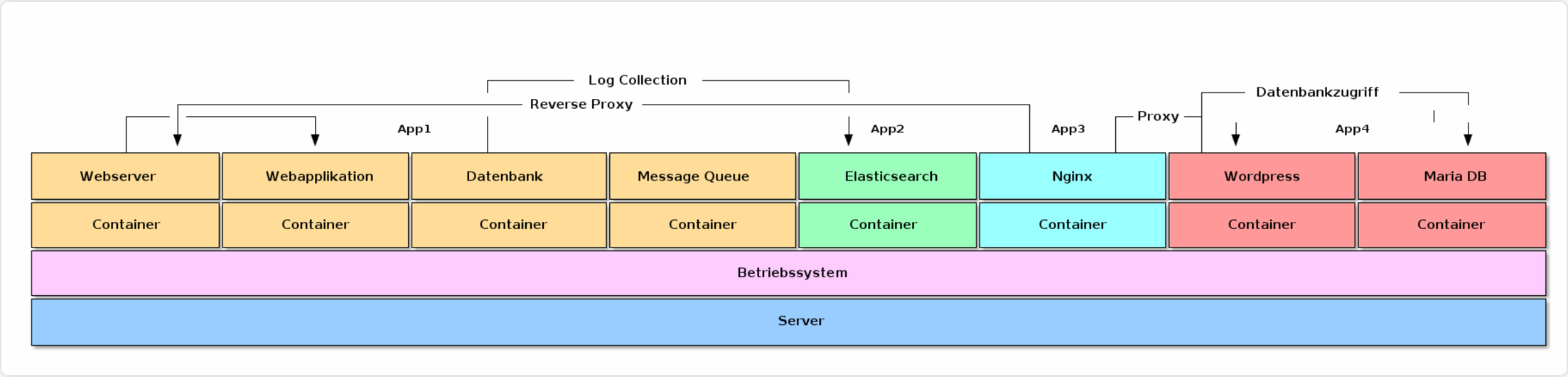
- Vom monolithischen Ansatz zum Microservice
- Pro Prozess ein Container

Separierung



- Ein Container pro Prozess
- Vereinfachtes Update
  - Definierte Kommunikation

Abschottung



- Verschiedene Stufen von Separierung
  - Container Apps die nur innerhalb ihrer Applikation kommunizieren
  - Apps die von allen Containern angesprochen werden
    - Syslog, Elasticsearch

Container und virtuelle Maschinen

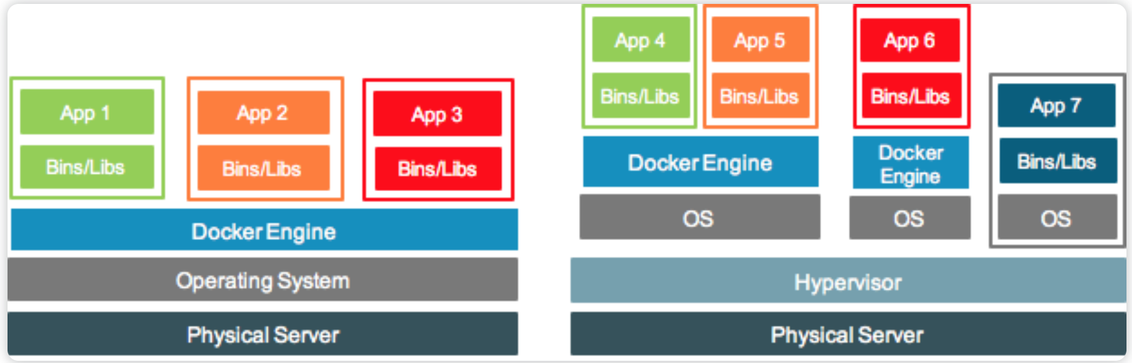
- Container
  - nativ Linux (teilt sich den Kernel)
  - geringer Footprint
- VM
  - komplettes Betriebssystem notwendig
  - Hardwareemulation

Warum betrifft uns das überhaupt?

- Offizielles Announcement von IBM
  - Docker support für Domino 9.0.1FP10
  - https://www-01.ibm.com/support/docview.wss?uid=swg22013200
- HCL Annoucement bei Fabric Tour 2
  - Neues Sametime auf Docker Basis
- Connections (ComponentPack)
  - AddOns als Container auf Basis von Kubernetes

VM oder Docker

- Sie unterscheiden sich, schliessen sich aber nicht aus



- Weniger Betriebssysteme und Machinen
  - Patch
  - Update
  - Support
- Geringere Lizenzkosten

Docker

Build once, run everywhere

Deploy everything nearly everywhere reliably and consistently.

- Bare Metal
- Virtuelle Server
- Cloud
- Bewahrt vor Installations-Albträumen
  - "Es lief auf meiner Maschine!"
  - Worked fine in Dev

Hauptvorteile

- Geschwindigkeit
  - Betriebssystem muss nicht gestartet werden
  - Online in Sekunden
- Portabilität
  - Weniger Abhängigkeiten zwischen den Prozessen
  - Verschieben zwischen verschiedenen Servern
- Effizienz
  - weniger Betriebssystem Overhead
  - verbesserte Resourcennutzung

Images und Container

- Ein Container wird durch Ausführen eines Images gestartet.
- Ein Image ist ein ausführbares Paket
  - enthält alles, was zum Ausführen einer Anwendung benötigt wird
  - Code, eine Laufzeit, Bibliotheken, Umgebungsvariablen und Konfigurationsdateien
- Container ist eine Laufzeitinstanz eines Images
- Liste Ihrer laufenden Container
  - `docker ps`
  - `docker container ls`

Docker Konzept

- Plattform für Entwickler und Administratoren
- Entwickeln, installieren und ausführen von Applikationen
- Containerization = Applikationen mit Containern ausführen
- Container sind nicht neu
  - die einfache Installation und Nutzung schon

Start eines Containers

```
➜ docker run alpine          (1)
➜ docker run -d alpine       (2)
➜ docker run -it alpine ash  (3)
```

1. Starte Container auf Basis Image `alpine`
2. Starte Container auf Basis Image `alpine` im Hintergrund
3. Starte interaktiv mit Pseudo TTY

- Welcher Container läuft noch?
- Woher kommen die Images
  - Registry
  - `hub.docker.com`

```
➜ docker ps
CONTAINER ID   IMAGE  COMMAND   CREATED         STATUS        PORTS  NAMES
cc1bc10a9091   alpine "ash"     9 seconds ago   Up 8 seconds         inspiring_greider
```

Gestoppte Container

- Gestoppte Container kann man mit `docker start <container>` starten
- Jeder `docker run` erstellt einen Container

```
➜ docker container ls -a
➜ docker ps -a

CONTAINER ID   IMAGE  COMMAND  CREATED         STATUS        PORTS  NAMES
cc1bc10a9091   alpine "ash"    9 seconds ago   Up 8 seconds         inspiring_greider
... lange Liste ...
```

- `docker run --rm` löscht den Container nach Beenden
- Sehr praktisch für lokale Tools

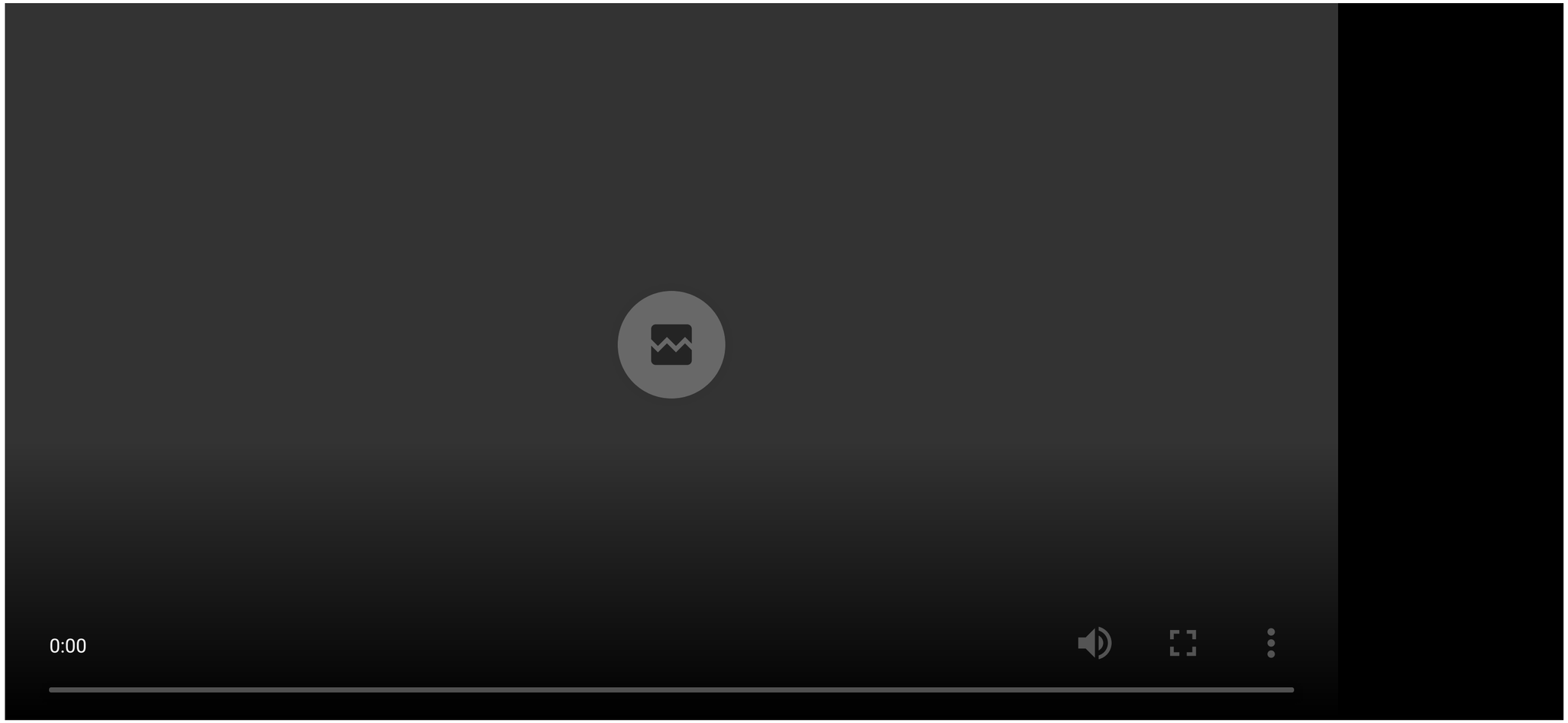
Docker Hub

- Sammlung aller möglichen Images
- `Dockerfile` für Ideen
- Bunte Mischung von Tools für lokale Ausführung und Server
  - WebSphere
  - Asciidoctor
  - curl

Offizielle und inoffizielle Images gemischt - Security?

Wie bekommt man ein neues Image

- Ändern eines interaktiv ausgeführten Containers



Besser ein Dockerfile

```
FROM alpine
RUN echo "Hello Admincamp" > test
CMD ["/bin/cat", "test"]
```

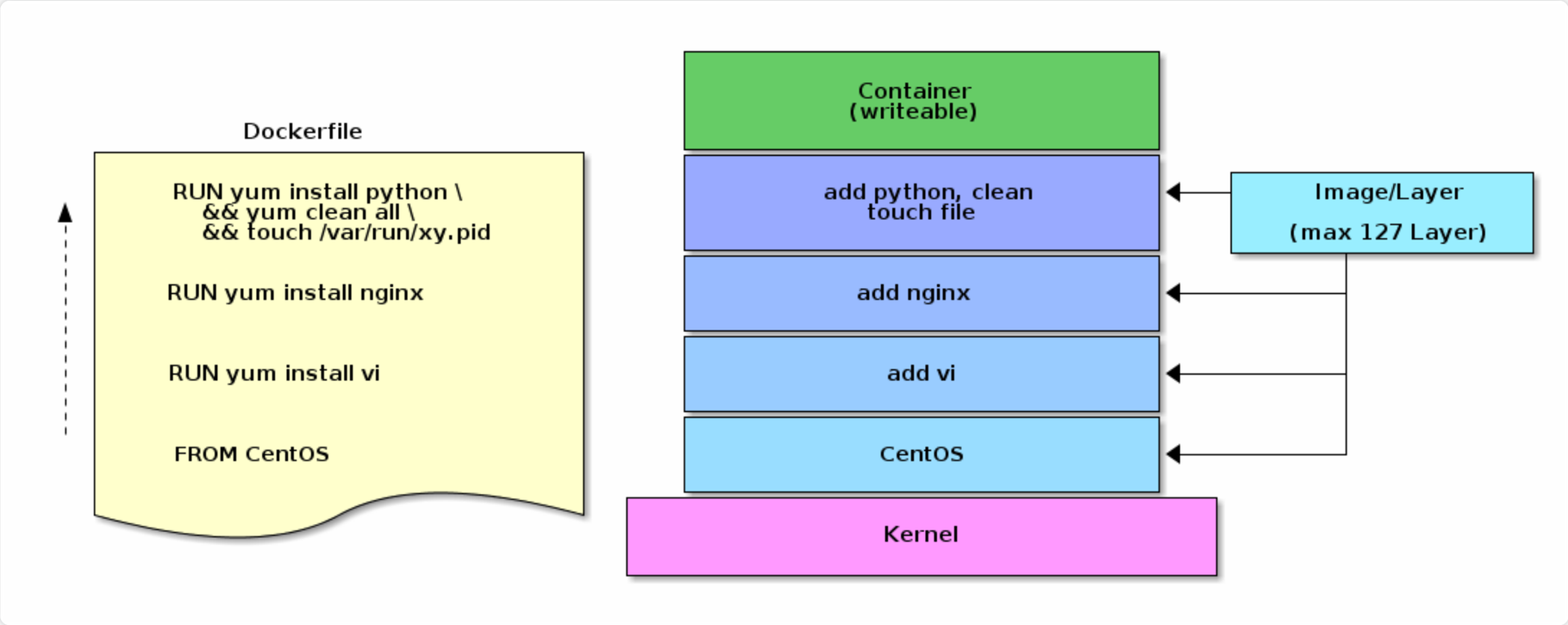
```
➜ docker build -t admincamp/hello hello
Sending build context to Docker daemon  2.048kB
Step 1/3 : FROM alpine
 ---> 196d12cf6ab1
Step 2/3 : RUN echo "Hello Admincamp" > test
 ---> Using cache
 ---> fc47f9b1cfcc
Step 3/3 : CMD ["/bin/cat", "test"]
 ---> Running in 91018a5e2f32
Removing intermediate container 91018a5e2f32
 ---> 915fa307573f
Successfully built 915fa307573f
Successfully tagged admincamp/hello:latest

➜ docker run admincamp/hello
Hello Admincamp
```

Vorteile bei der Verwendung von Dockerfiles

panagenda

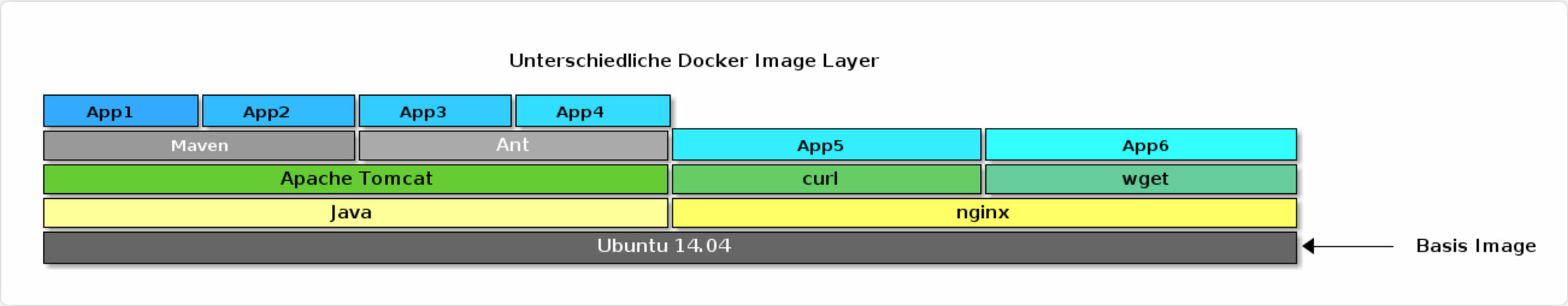
- **Wiederholbare** Ergebnisse
- `Dockerfile` kann in Versionsverwaltung abgelegt werden
- Image immer wieder gleich und neu gebaut
- Alternativ `docker-compose`
  - Bau von abhängigen Images
  - z.B. Wordpress + MySQL in zwei getrennten Images
  - YAML

Image Layer



- Image ist eine Sammlung Dateien
- Layer sind Images und readonly
- Cow Mechanismus (Copy-on-write)

Layer teilen



- Images können Layer teilen/gemeinsam nutzen
  - Beschleunigt Übertragungszeiten
  - Optimiert Disk- und Speichernutzung
- Bereits vorhandene Images müssen nicht mehr heruntergeladen werden

Einfacher Container

- Download letztes Alpine image
  - Update
  - `curl` installieren

Dockerfile alpine

```
FROM alpine:latest
RUN apk update
RUN apk upgrade
```

Dockerfile curl

```
FROM alpine:latest
RUN apk update
RUN apk upgrade
RUN apk add curl
```

```
➜ docker build -t admincamp/alpine alpine
➜ docker build -t admincamp/alpine curl
```

Einfacher Container und Layer

alpine

```
➜ docker history admincamp/alpine:latest
IMAGE               CREATED             CREATED BY                                      SIZE
36600b1d7064        2 minutes ago       /bin/sh -c apk upgrade                          20.6kB
c3a7f6b38c2a        2 minutes ago       /bin/sh -c apk update                           1.28MB
196d12cf6ab1        2 days ago          /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>           2 days ago          /bin/sh -c #(nop) ADD file:25c10b1d1b41d46a1…   4.41MB
```

curl

```
➜ docker history admincamp/curl:latest
IMAGE               CREATED             CREATED BY                                      SIZE
445b6b03a6f5        10 seconds ago      /bin/sh -c apk add curl                         1.52MB (1)
36600b1d7064        2 minutes ago       /bin/sh -c apk upgrade                          20.6kB
c3a7f6b38c2a        2 minutes ago       /bin/sh -c apk update                           1.28MB
196d12cf6ab1        2 days ago          /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>           2 days ago          /bin/sh -c #(nop) ADD file:25c10b1d1b41d46a1…   4.41MB
```

(1) Zusätzlicher Layer

Dockerfile optimieren

Dockerfile vim-nox

```
FROM ubuntu:18.04
ENV DEBIAN_FRONTEND=noninteractive
ENV DEBCONF_NONINTERACTIVE_SEEN=true
RUN apt-get update
RUN apt-get install -y --no-install-recommends vim-nox
RUN rm -rf /var/lib/apt/lists/*
```

Dockerfile vim-nox optimized

```
FROM ubuntu:18.04
ENV DEBIAN_FRONTEND=noninteractive
ENV DEBCONF_NONINTERACTIVE_SEEN=true
RUN apt-get update \
    && apt-get install -y --no-install-recommends vim-nox \
    && rm -rf /var/lib/apt/lists/*
```

```
➜ docker image ls
REPOSITORY                TAG     IMAGE ID      CREATED         SIZE
admincamp/vim-optimized   latest  d5d59d05f003  4 minutes ago   201MB
admincamp/vim             latest  81ff3f410667  5 minutes ago   242MB
```

Woher kommt der Größenunterschied?

- Download der apt Repositories = 42 MB

```
➜ docker history admincamp/vim:latest
IMAGE               CREATED             CREATED BY                                      SIZE
81ff3f410667        6 minutes ago       /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
d917cdf69ca1        6 minutes ago       /bin/sh -c apt-get install -y --no-install-r…   116MB
1ee1bc3260e5        6 minutes ago       /bin/sh -c apt-get update                       41.6MB
6bf8b9ae7f9f        6 minutes ago       /bin/sh -c #(nop)  ENV DEBCONF_NONINTERACTIV…   0B
```

- Layer sind readonly
  - Daten die wieder gelöscht werden in einem RUN laden und löschen

```
➜ docker history admincamp/vim-optimized:latest
IMAGE               CREATED             CREATED BY                                      SIZE
d5d59d05f003        5 minutes ago       /bin/sh -c apt-get update     && apt-get ins…   116MB
6bf8b9ae7f9f        6 minutes ago       /bin/sh -c #(nop)  ENV DEBCONF_NONINTERACTIV…   0B
```

Wichtige Kommandos

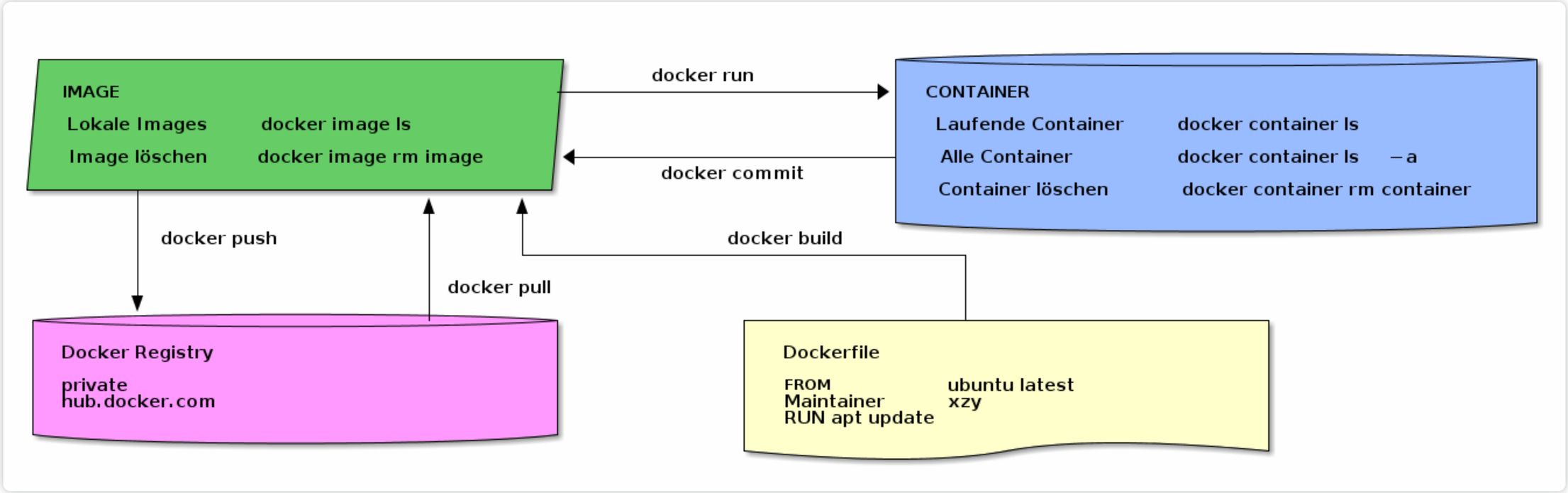


Image suchen

```
➜ docker search websphere
NAME                              DESCRIPTION                                     STARS   OFFICIAL
websphere-liberty                 Official IBM WebSphere Application Server …   202     [OK]
ibmcom/websphere-traditional      Official IBM WebSphere Application Server …   103
ibmcom/websphere-liberty          Official IBM WebSphere Application Server …   40
amanly/websphere_8_5_5            This container has WebSphere installed.       20
...
```

- Durchsucht Docker Hub
- Anzeige "Stars" & Official Tag

Download Image

```
➜ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
4fe2ade4980c: Pull complete
Digest: sha256:621c2f39f8133acb8e64023a94dbdf0d5ca81896102b9e57c0dc184cadaf5528
Status: Downloaded newer image for alpine:latest
```

```
➜ docker pull ibmcom/websphere-traditional
Using default tag: latest
latest: Pulling from ibmcom/websphere-traditional
3b37166ec614: Pull complete
ba077e1ddb3a: Pull complete
34c83d2bc656: Pull complete
84b69b6e4743: Pull complete
0f72e97e1f61: Pull complete
4c5dc39ba08b: Pull complete
646f7e8cfada: Pull complete
2bf0d2c7b514: Pull complete
a58cfa6db9a7: Pull complete
Digest: sha256:71a8f0386ea4aab437cad990f4d22b05b5caeab6744d817876be0f6c303a19c3
Status: Downloaded newer image for ibmcom/websphere-traditional:latest
```

Lokale Images anzeigen

```
➜ docker image ls
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
alpine                              latest              196d12cf6ab1        44 hours ago        4.41MB
ibmcom/websphere-traditional        latest              f786ba8916fb        7 days ago          1.76GB
```

- Hier sammelt sich nach kurzer Zeit eine Menge Kram
  - Images, Layer etc
- Aufräumen, ab Docker 1.25+ mit `docker prune`

```
➜ docker  image prune [OPTIONS]
```

Option	Beschreibung
--all, -a	Löscht alle unbenutzten Images, nicht nur ungetaggte
--filter	Filter für zu löschende Images
--force, -f	Kein Sicherheitsprompt

Daten in Container kopieren

- Analog `Dockerfile` Kommando `COPY|ADD`

```
➜ docker run -name nginx-copy nginx
➜ docker container cp index.html container:/usr/share/nginx/html
```

- Nett für Test, aber sollte man automatisieren
- Kann man in Verbindung mit `docker commit` für erste Versuche verwenden

Praktisch noch nie verwendet

Lokalen Ordner mappen

```
➜ docker container run -v $(pwd)/nginx:/usr/share/nginx/html/ -d -p 8080:80 nginx
```

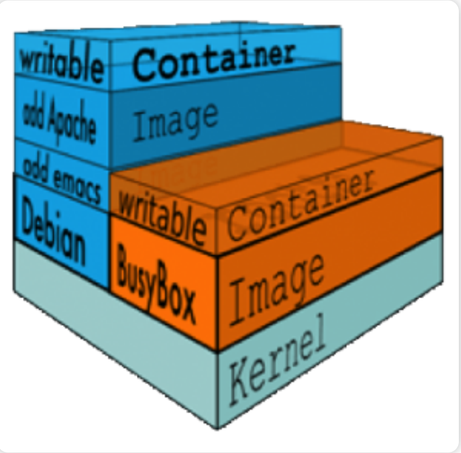
- `$(pwd)` gibt den aktuellen Pfad aus
- Map des lokalen Ordners …/`nginx` in den `html` Ordner von nginx
- Es wird nur der Ordner-Inhalt in den Container verbunden
  - Dateien des Containers die in diesem Pfad liegen werden ignoriert
  - Achtung bei `config`-Ordnern → alle Dateien bereitstellen

Docker Volumes

panagenda

> Volumes haben gleichen Inhalt wie Originalordner

- Speichern von Daten über die Lebenszeit eines Containers hinaus




```
➜ docker container run -it -v /usr/share/nginx/html nginx
a8e549da56018462b16d95ee646a88077759e63c27a1fa2bbc4208c34cc6d85d

➜ docker volume create --name nginx1
➜ docker container run -it -v nginx1:/usr/share/nginx/html nginx
nginx1
```

Volumes finden

- Überprüfen welche Volumes vom Container verwendet werden

```
➜ docker inspect -f "{{json .Mounts}}"  a8e549da5601 | jq .
[
  {
    "Type": "volume",
    "Name": "e06a4d29ab94665c52bc6ae2c1fbb281b3501c5068cba19ca736ad662b242423",
    "Source": "/var/lib/docker/volumes/e06a4d29a...01c5068cba19ca736ad662b242423/_data",
    "Destination": "/usr/share/nginx/html",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
]
```

- das manuell erstellte Volume heißt `nginx1`
- `Name`, `Source` und `Destination` verweisen auf den Namen

Volumes anderer Container

```
➜ docker run -it --volumes-from nginx1 alpine ash
```

- direkten Zugriff auf das Volume
- Unabhängig ob der Original-Container läuft
- Praktisch verwendet für
  - Updates
  - Backup
  - Datenbank-Troubleshooting
    - Container mit Admintool erstellen
    - Volume mounten

Praktischer Einsatz

- `nginx` mit ihrem Webserver
  - Datencontainer für html Seiten
- Neue Version von nginx mit gleichen html Seiten

```
➜ docker volume create vol-nginx-web
➜ docker run -d --name nginx-web -v vol-nginx-web:/usr/share/nginx/html nginx:1.14.0-alpine
➜ docker container cp index.html nginx-web:/usr/share/nginx/html

➜ docker stop nginx-web

➜ docker run -d --name nginx-web-neu -v nginx-web:/usr/share/nginx/html nginx:1.15.0-alpine
```

- Oder Verwendung mit `:latest`, dann regelmäßig `docker pull`

Troubleshooting

- `docker exec`

```
# Start bash in einem laufenden Container
docker exec <containername> /bin/bash
```

- `docker inspect <containername>`
  - Zeigt Informationen zum Container (Ports, Volumes)
- `docker logs <containername>`

Option	Beschreibung
--follow , -f	Log Ausgabe folgen
--tail n	die letzten n Zeilen des Logs
--until	Zeitstempel oder Zeit (42min)

Aufräumen

- `docker image prune` kennen wir schon

```
➜ docker container prune
```

- Löscht alle gestoppten Container

```
➜ docker volume prune
```

- Löscht nicht mehr verwendete Volumes *

Mit `volume prune` sehr vorsichtig sein!

`docker run --rm` löscht Container beim Beenden

Alles auf einmal

```
➜ docker system prune

WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all dangling images
        - all build cache
Are you sure you want to continue? [y/N] y
```

Option	Beschreibung
--all, -a	Alle unbenutzten Images löschen
--filter	Filtern nach z.B. label=
--force, -f	Den Bestätigungsdialog nicht anzeigen
--volumes	Auch alle Volumes löschen

Netzwerk

Container Netzwerk

- Docker Netzwerk basiert auf `iptables`
- Device: `docker0`

```
➜ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4ed464c0dee8        bridge              bridge              local
4b1297dbc7ff        host                host                local
ccd0dbe49ed6        none                null                local
```

- Benutzerdefinierte Netzwerke
  - Bessere Kontrolle der Container Kommunikation
  - DNS der Container-Namen

Netzwerk Ports veröffentlichen

Flag	Beschreibung
-p 8080:80	TCP Port 80 des Containers zum Port 8080 des Docker Hosts
-p 192.168.1.100:8080:80	TCP Port 80 des Containers zum Port 8080 des Docker Hosts auf IP 192.168.1.100
-p 8080:80/udp	UDP Port 80 des Container zum Port 8080 des Docker Hosts
-p 8080:80/tcp -p 8080:80/udp	TCP Port 80 des Containers zum TCP Port 8080 Docker Host UDP Port 80 des Containers zum UDP port 8080 Docker Host

Netzwerk Ports Beispiele

```
➜ docker run -d --name nginx-test1 stoeps1 -p 8080:80 nginx

➜ docker port nginx-test1
80/tcp -> 0.0.0.0:8080

➜ docker run -d --name nginx-test2 stoeps2 -p 192.168.0.192:8081:80 nginx

➜ docker port nginx-test2
80/tcp -> 192.168.0.192:8081

➜ docker container ls
CONTAINER ID   IMAGE   COMMAND                  ...   PORTS                          NAMES
36f4c84a84d9   nginx   "nginx -g 'daemon of…"   ...   192.168.0.192:8081->80/tcp   nginx-test2
a35bbd79bf30   nginx   "nginx -g 'daemon of…"   ...   0.0.0.0:8080->80/tcp         nginx-test1
```

- `nginx-test2` ist nicht auf `Localhost` erreichbar

Ports in Dockerfiles

```
EXPOSE <port> [<port>/<protocol>...]

# z.B.
EXPOSE 80
```

- Image enthält dann bereits die Information des Ports
- EXPOSE gibt den Port nur innerhalb des Docker Netzwerks frei
- Zusätzlich `-p 8080:80` notwendig

```
➜ docker run -d --name nginx-test3 -P nginx

➜ docker port nginx-test3
80/tcp -> 0.0.0.0:32768
```

- `-P` veröffentlich alle im Image verwendeten Ports (dynamische Portvergabe am Host)

DNS

- Container lesen **nicht** die lokale `/etc/hosts`!
  - Link auf lokale `/etc/resolv.conf`
- DNS Auflösung notwendig
  - z.B. Container mit `bind`

Flag	Description
--dns	IP Adresse eines DNS Servers
--dns-search	Search Domain `example.com`
--dns-opt	Optionen für `resolv.conf`, z.B. `timeout:1`
--hostname	Hostname des Containers

```
docker run --dns 10.10.10.10 --dns 192.168.1.2 --dns-search example.com --dns-search example.org ...
```

Upstream Proxy verwenden

- Firmenumgebungen erlauben keinen direkten Internetzugriff
- < Docker 17.06
  - Umgebungsvariablen in jedem Container
    - `ENV HTTP_PROXY "http://proxy:3128"`
    - Container nicht mehr portabel
  - `/etc/default/docker`
    - `export http_proxy="http://127.0.0.1:3128/"`

Upstream Proxy in neuen Docker Versionen

- >= 17.06
  - ~/.config.js

```
{
  "proxies":
  {
    "default":
    {
      "httpProxy": "http://127.0.0.1:3001",
      "noProxy": "*.test.example.com,.example2.com"
    }
  }
}
```

Docker-Compose

- Verwaltet Applikationen aus mehreren Containern
- Gibt Ports für die interne Kommunikation frei
- Startet, Stoppt und Zerstört die Container
- Command-line:
  - `docker-compose`
  - YAML Datei `docker-compose.yml`
- Verwendet den Ordnernamen zur Abgrenzung
  - Mehrere isolierte Anwendungen auf einem Host möglich

Beispiel Wordpress mit MySQL

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
```

Beispiel Loganalyse mit ELK

```
elasticsearch:
  image: elasticsearch:6.4.2
  ports:
    - "9200:9200"
    - "9300:9300"
  volumes:
    - ./es-data:/usr/share/elasticsearch/data
  environment:
    ES_JAVA_OPTS: "-Xmx512m -Xms512m"
logstash:
  image: stoeps/logstash:5.6.13
  command: logstash --debug -f /usr/share/logstash/pipeline
  volumes:
    - ./conf.d:/usr/share/logstash/pipeline
    - ./logs:/opt/logs
  ports:
    - "5000:5000"
  links:
    - elasticsearch
kibana:
  image: kibana:6.4.2
```

Orchestrierung

Kubernetes und Konsorten

- Docker läuft auf einem Host → keine Redundanz
- Orchestrierung notwendig
  - Kubernetes (ursprünglich von Google)
  - Docker Swarm
- Kubernetes offiziell von Docker unterstützt

Kubernetes

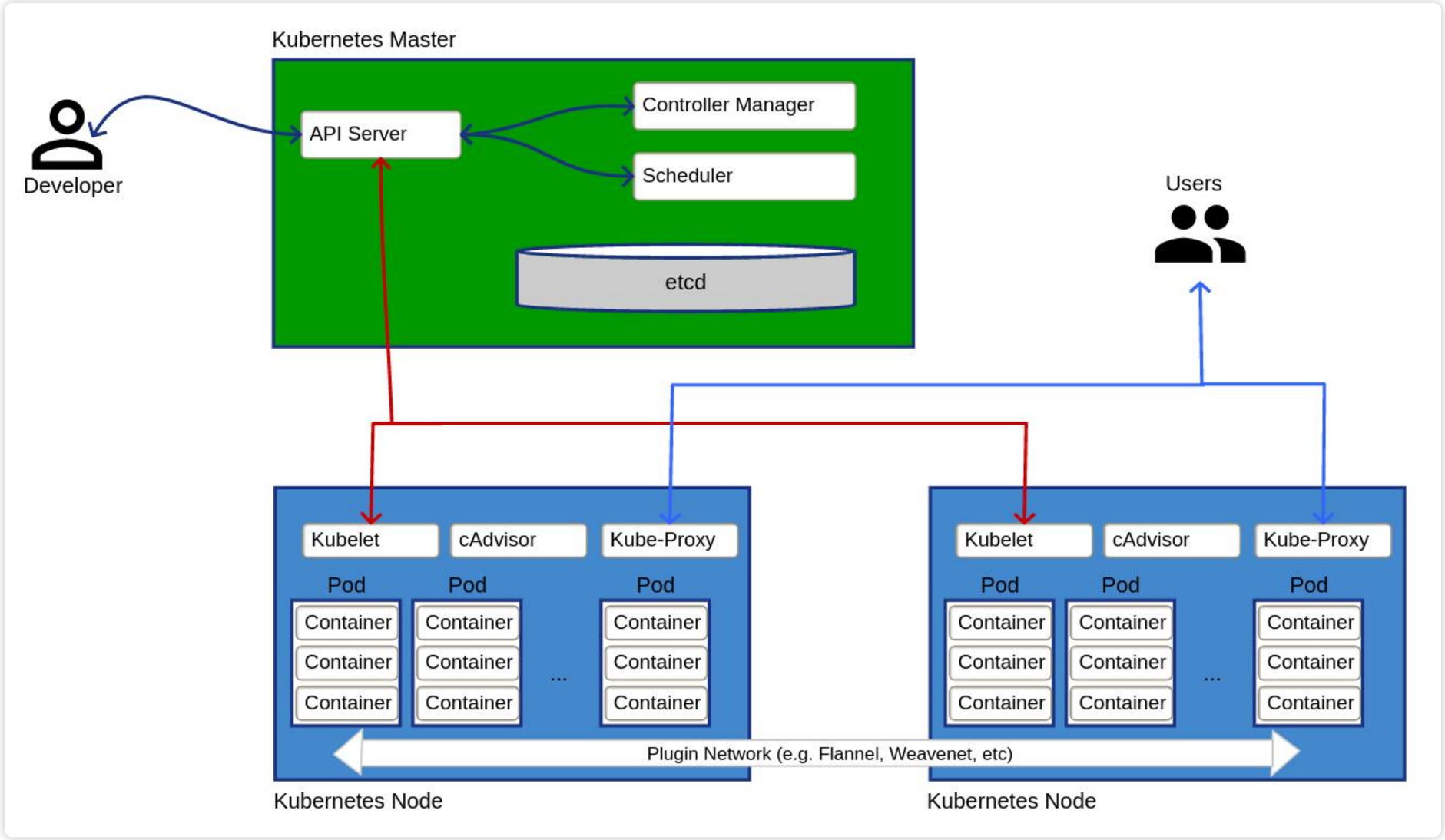
- Hardware
  - Node (Einzelmaschine)
  - Cluster (Zusammenfassung von Nodes)
  - Persistent Volume (Netzwerk-Storage)
- Software
  - Container (Docker)
  - Pods (Container mit gleichen Resourcen)
  - Deployments
  - Ingress

Analog zur Docker CLI gibt es `kubectl`

Pods

- ein oder eine Gruppe von mehreren Containern
- auf einem einzelnen Node implementiert
- Container in einem Pod teilen sich
  - IP-Adresse
  - IPC
  - Hostname
  - weitere Ressourcen
- Netzwerk und Storage vom Container
  - Einfacheres Clustermanagement

Kubernetes Übersicht



Container Management

- Liveness Checks
  - Container starten automatisch neu
- Abhängigkeiten mehrerer Container in einem Pod
- Unterstützt mehrere Nodes (Failover, Lastverteilung)
  - Beispiel: IBM Cloud Private unterstützt 1000 Nodes

Kubernetes kann Non-HTTP Traffic nur über Workaround veröffentlichen

- https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/exposing-tcp-udp-services.md

Vorteile von Docker

Unresolved directive in AC19-T2S1-DockerWasIstDasEigentlich.adoc - include::slide{}[]

- `Dockerfile` und `docker-compose.yml` versionierbar
  - Oft als Dokumentation vollkommen ausreichend
  - Gut integrierbar in Dokumente auf `Markdown` oder `Asciidoctor` Basis
- Erfodert ein Umdenken
  - Kein Troubleshooting im Container
  - Keine Konfigurationsänderungen in laufender Applikation
  - Erstellen eines neuen Containers
  - Tausch der Applikation

Learning Resources

+49 173 8588719
christoph.stoettner@panagenda.com
linkedin.com/in/christophstoettner
stoeps.de
christophstoettner
@stoeps

IBM CHAMPION

THANK YOU!