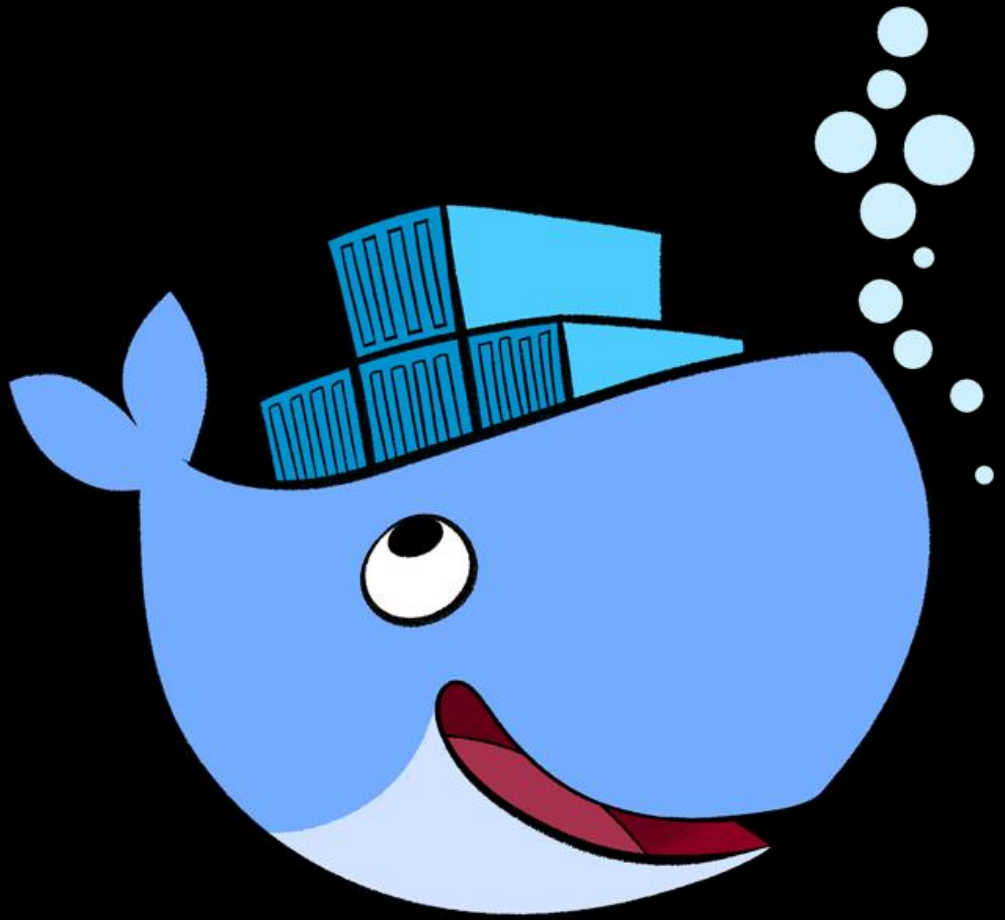


Docker

Grundlagen Workshop

Trier Tech Talk Conference, 29.04.2017

Nico Maas

Zur Person

Nico Maas

IT Systemelektroniker

Bachelor of Science

mail@nico-maas.de

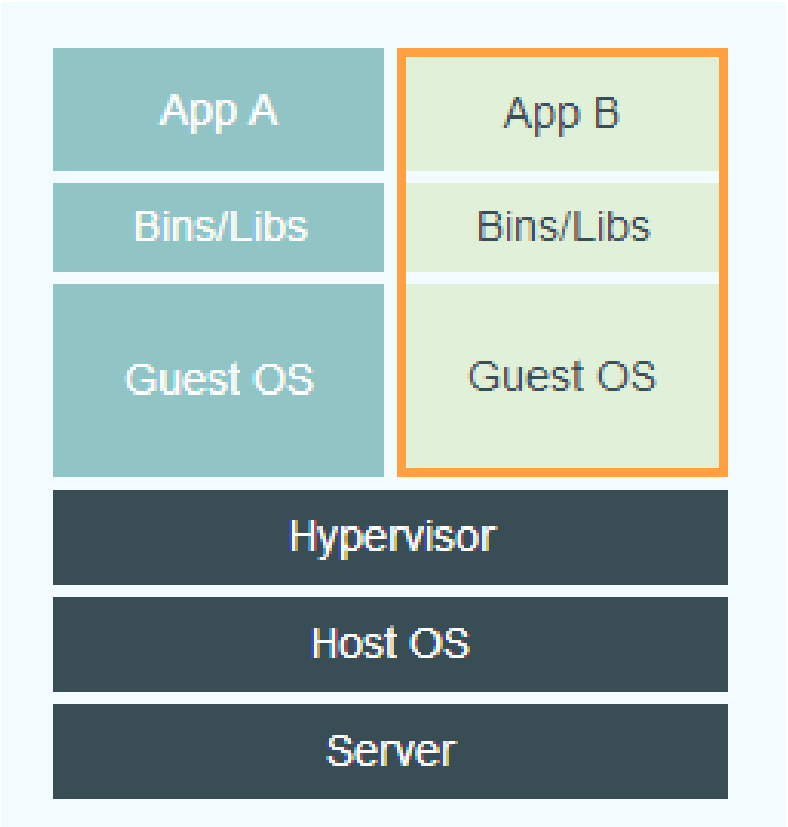
www.nico-maas.de

@nmaas87

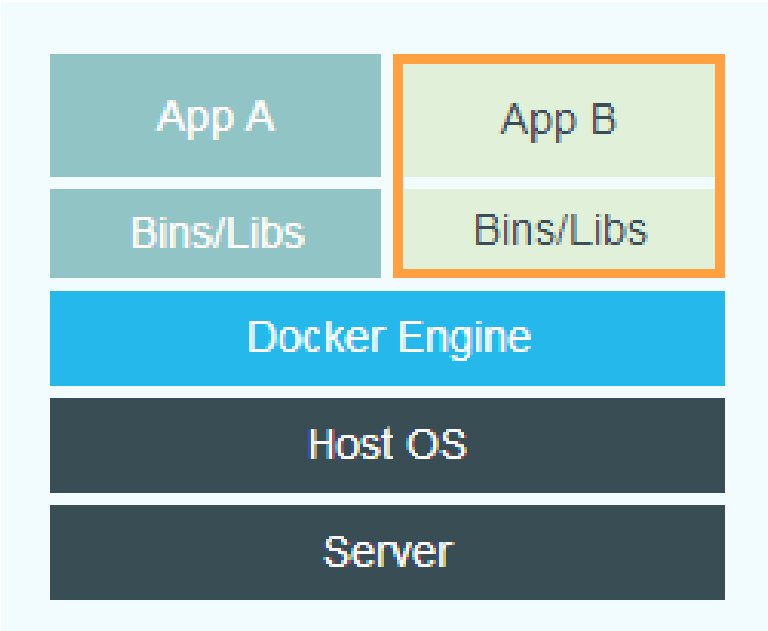
Agenda

- **I. Einführung**
  - VMs vs Container
  - Container Technologie
  - Docker Architektur
  - Images / Container
- **II. Workshop**
  - Demo Docker Grundlagen
  - Demo Dockerfile
  - docker-compose
  - Docker Swarm
- **III. Docker im Einsatz**
  - resin.io
  - gogs.io
  - drone.io

I. VMs vs Container



Docker



[Docker]

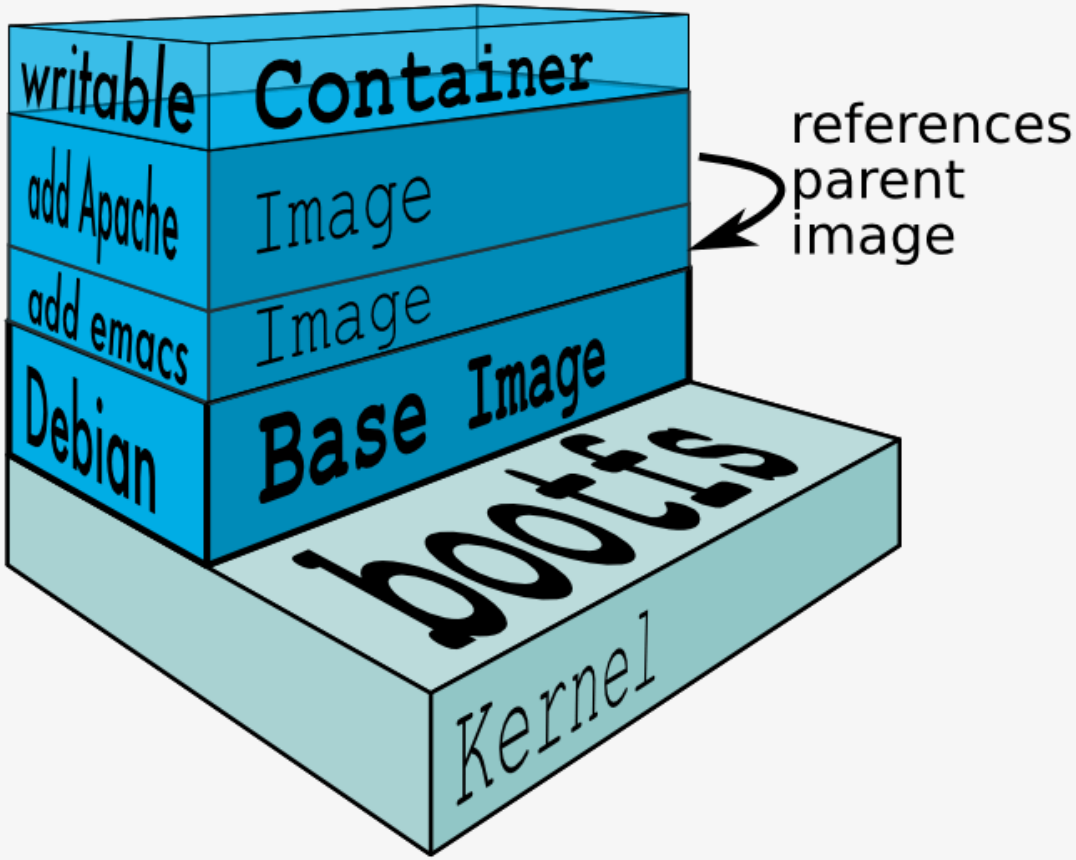
I. Matrix From Hell

The Matrix of Hell

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

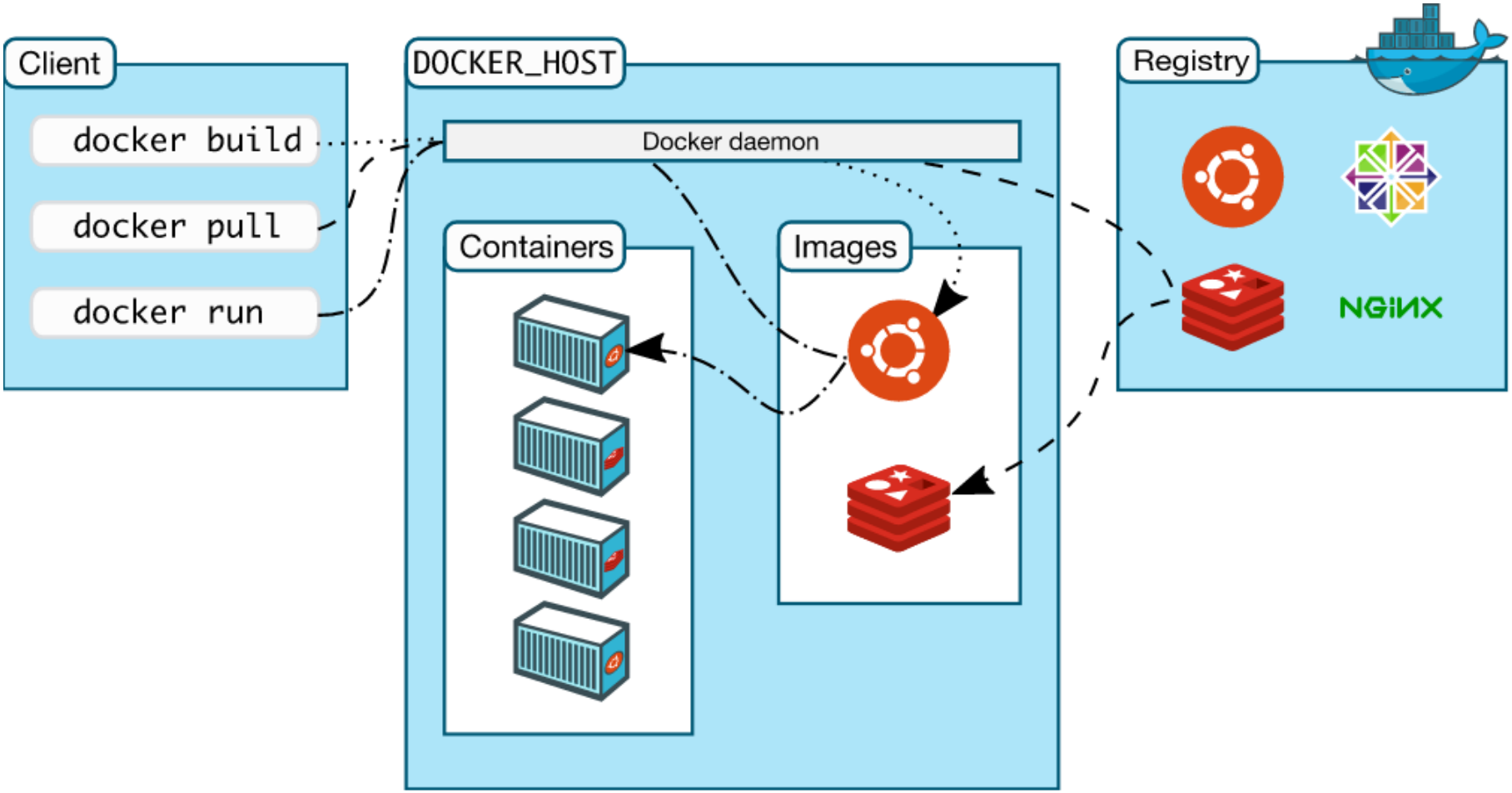
https://blog.docker.com/2013/08/paas-present-and-future/

I. Container



http://stackoverflow.com/questions/24702233/docker-container-and-memory-consumption

I. Docker Architektur



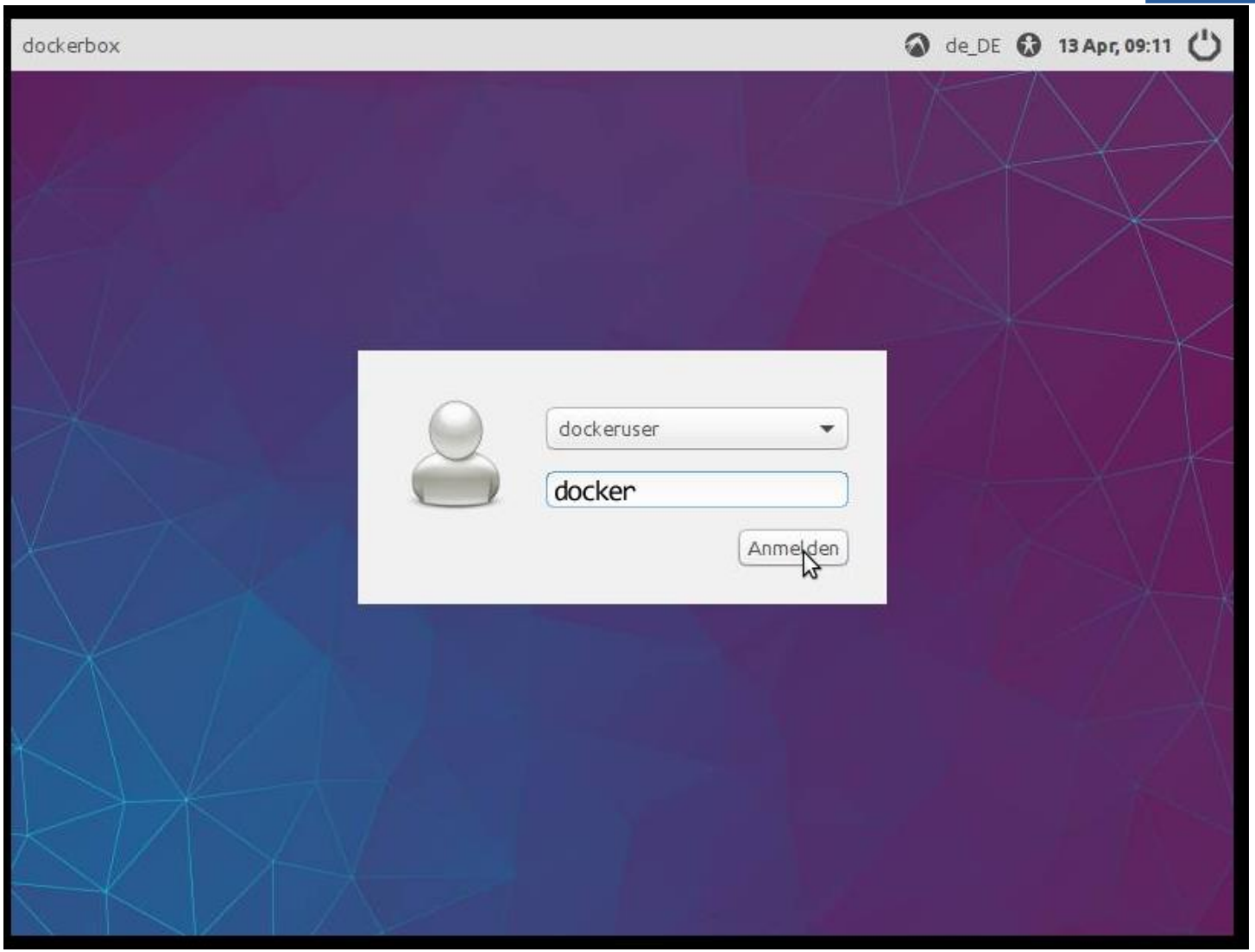
I. Image / Container

- Images
  - Dateisystem
  - R/O
  - Ein ganzes Repo gibt es auf hub.docker.com
- Container
  - laufende Instanz eines Images
  - R/W
  - Änderungen müssen mit docker commit „gespeichert werden“ damit sie permanent werden und zu einem neuen Image

I. Aufbau Images

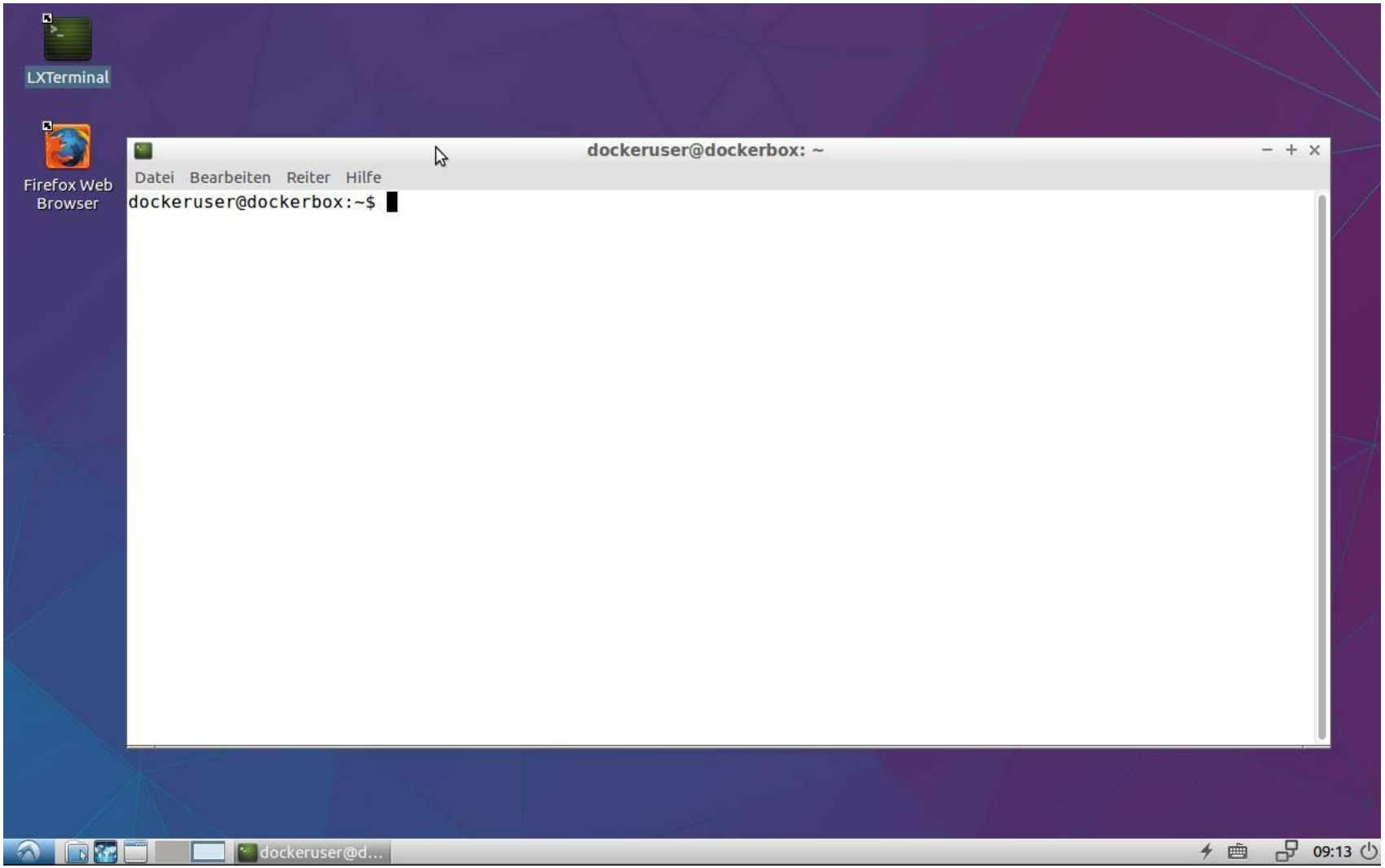
- Images bestehen aus Layern
- Neue Images am Besten nicht von Hand, sondern per Dockerfile bauen
- Jeder Befehl in einer Dockerfile wird ein eigener Layer
- Möglichst wichtige Befehle / Basis direkt am Anfang und zusammenfassen damit das eigene Caching von Docker besser funktioniert – häufig sich ändernde Dateien ans Ende
- Beispiel für ein Layer: https://imagelayers.io/?images=microscaling%2Fimagelayers-api:latest,microscaling%2Fimagelayers-web:latest

II. Demo Docker Grundlagen



II. Demo Docker Grundlagen

II. Demo Docker Grundlagen



II. Demo Docker Grundlagen

- docker
  - Ausgabe Dockerhilfe
- docker version
  - Versionsausgabe
- docker ps
  - Laufende container Anzeigen
- docker ps –a
  - Gestoppte Container anzeigen
- docker run nmaas87/docker-openwrt echo "Hallo T3C 2017!"
  - Image nmaas87/docker-openwert auf dem „latest“ Stand von hub.docker.com herunterladen, starten und dort den Befehl echo "Hallo T3C 2017!" ausführen

II. Demo Docker Grundlagen

- docker run -d nmaas87/docker-openwrt ping 127.0.0.1 -c 50
  - ping Befehl im deattached Mode ausführen
- docker ps
  - Laufende Container anzeigen lassen, z.B.

```
PS C:\Users\Nico Maas> docker ps
CONTAINER ID        IMAGE                   COMMAND                  CREATED             STATUS              PORTS               NAMES
a9472cce6998        nmaas87/docker-openwrt  "ping 127.0.0.1 -c 50"   3 seconds ago       Up 1 seconds                            determined_hoover
```

- docker attach a94
  - Wieder mit dem laufenden Container verbinden

- Container ”stirbt” automatisch wenn der darin laufende Prozess (ping, läuft als Prozess ID 1) endet

II. Demo Docker Grundlagen

- docker run -it nmaas87/docker-openwrt /bin/sh
  - Interaktive Shell im Container starten
- STRG + P + Q
  - Von der Shell trennen
- docker ps
  - Laufende Container anzeigen lassen, z.B.

```
PS C:\Users\Nico Maas> docker ps
CONTAINER ID        IMAGE                   COMMAND             CREATED             STATUS              PORTS               NAMES
d9fea9bf9c27        nmaas87/docker-openwrt  "/bin/sh"           6 seconds ago       Up 5 seconds                            stupefied_newton
```

- docker attach d9f
  - Wieder mit dem laufenden Container verbinden

II. Demo Dockerfile

- Wird verwendet um ein neues Docker Image zu bauen / „Kochrezept“
- Dockerfile Beispiel (siehe: https://github.com/nmaas87/docker-demo)

```
FROM nmaas87/docker-openwrt:15.05.1_x86

# Update opkg Package List, Install python 2.7, pip, Flask and create directories
RUN opkg update && \
    opkg install wget ca-certificates libffi libopenssl python-openssl python && \
    cd /tmp && \
    wget https://bootstrap.pypa.io/get-pip.py && python get-pip.py && \
    pip install --no-cache-dir Flask && \
    mkdir /app/ /app/templates

# copy files required for the app to run
COPY app.py /app/
COPY templates/index.html /app/templates/

# tell the port number the container should expose
EXPOSE 80

# run the application
#CMD /sbin/init
CMD ["python", "/app/app.py"]
```

II. Demo Dockerfile

- git clone https://github.com/nmaas87/docker-demo.git
- cd docker-demo
- docker build –t app .
  - Erstelle das Image app aus der Dockerfile mit dem Namen „Dockerfile“ welche du im gleichen Verzeichnis (hier mit . beschrieben) gefunden hast
- docker run –p80:80 app
- Firefox starten und auf http://127.0.0.1 gehen ☺

- Ab jetzt können wir schnell die app.py ändern, das Image neu bauen und starten und damit den Software Entwicklungszyklus simulieren.

II. docker-compose

- Wird verwendet um leichter Docker Images auszuführen
- docker-compose.yml (siehe: https://github.com/nmaas87/docker-demo)

```
flaskapp:
  restart: unless-stopped
  image: app
  ports:
    - "80:80"
```

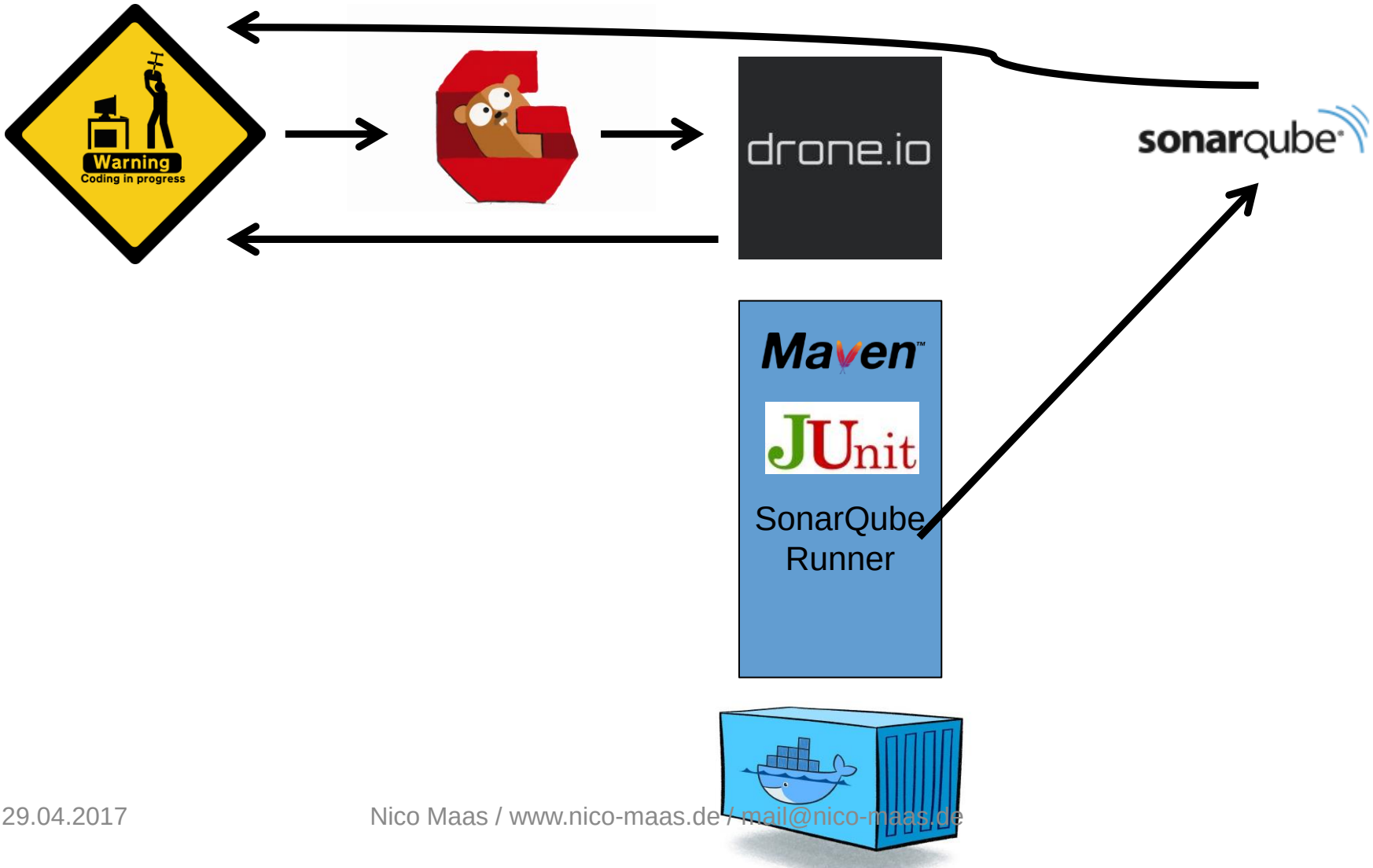
- „Erzeuge einen Container mit dem Namen flaskapp, welchen du immer wieder neustartest (selbst wenn der Service darin stirbt oder der Server neugestartet wird), verwende dazu das lokale Image app (eben gebaut) und binde den Host Port 80 an den Container Port 80 (HOST:Docker Container)
- Starten mit docker-compose up oder docker-compose up –d (für deattached Mode)
- Auf http://localhost gehen um die Webseite zu sehen ☺

II. Docker Swarm

- Redundantes Cluster von Docker Nodes, in gewissem Grade selbstheilend, mit Routing Overlay Network
- Swarm erstellen:
  - Auf dem "Master/Manager" Swarm Netzwerk erstellen:
    - `docker swarm init --advertise-addr <MANAGER-IP>`
    - Gibt anschließend ein CMD aus ähnlich:
    - `docker swarm join --token <token> <MANAGER-IP>:2377`
  - Dieses CMD auf allen Docker Nodes ausführen
  - fertig!
- Service erstellen
  - `docker service create --replicas 1 --name pinger nmaas87/docker-openwrt:trunk_x86 ping docker.com`
  - `docker service ls`
  - `docker service inspect --pretty pinger`
  - `docker service ps pinger`
  - `docker service scale pinger=3`
  - `docker service rm pinger`
- Nützliche CMDs (auf Manager!)
  - `docker node ls`
  - `docker info`

II. Demo Gogs / Drone / Maven / JUnit

- Live Demo



III. Docker im Einsatz

- www.resin.io
  - Verwendet Docker als Deployment Methode von Applikationen z.B. auf Raspberry Pi oder anderen ARM Plattformen
- www.gogs.io
  - Gitlab Clone welcher in Go geschrieben ist, als Docker Container gepackt
- www.drone.io
  - CI System welcher als Docker Image kommt und Docker Container verwendet um Testing zu verwenden, besonders gut in Zusammenarbeit mit Gogs

IV. Fragen?

Vielen Dank für Ihre Aufmerksamkeit!

V. Quellenangabe

- Grafik Frontfolie:
  - https://www.docker.com/products/docker-engine