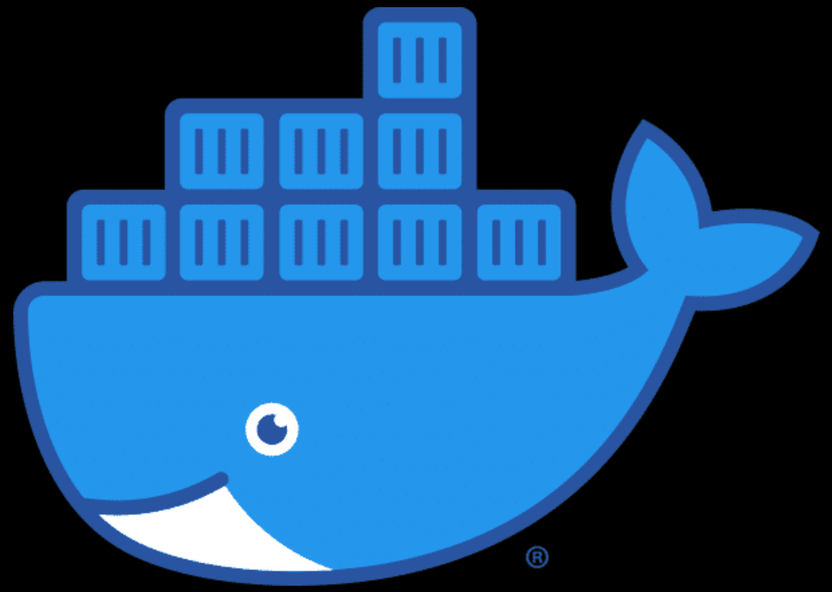
Build it.
Ship it.
docker run -it

Stefan Schöberl
schoeberl.dev

stefanschoeberl

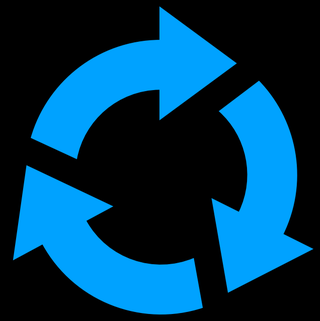
Workshop

- Wozu Docker?
- Container
- Images
- Dockerfile
- Multi-Stage Builds
- Docker Compose
- Tipps, Tricks & Nützliches

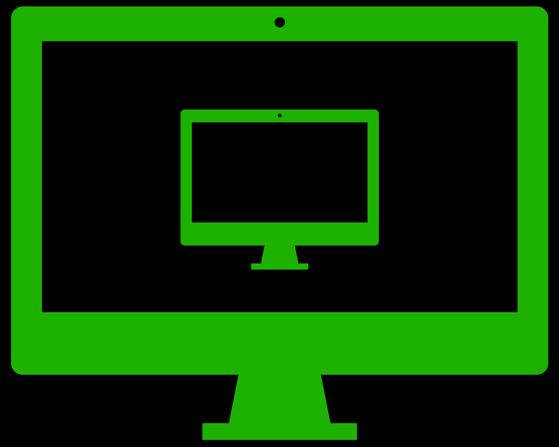


Wozu Docker?

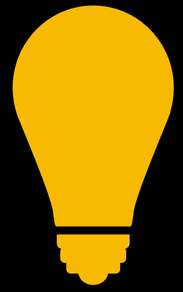
„It works on my machine“



CI / CD

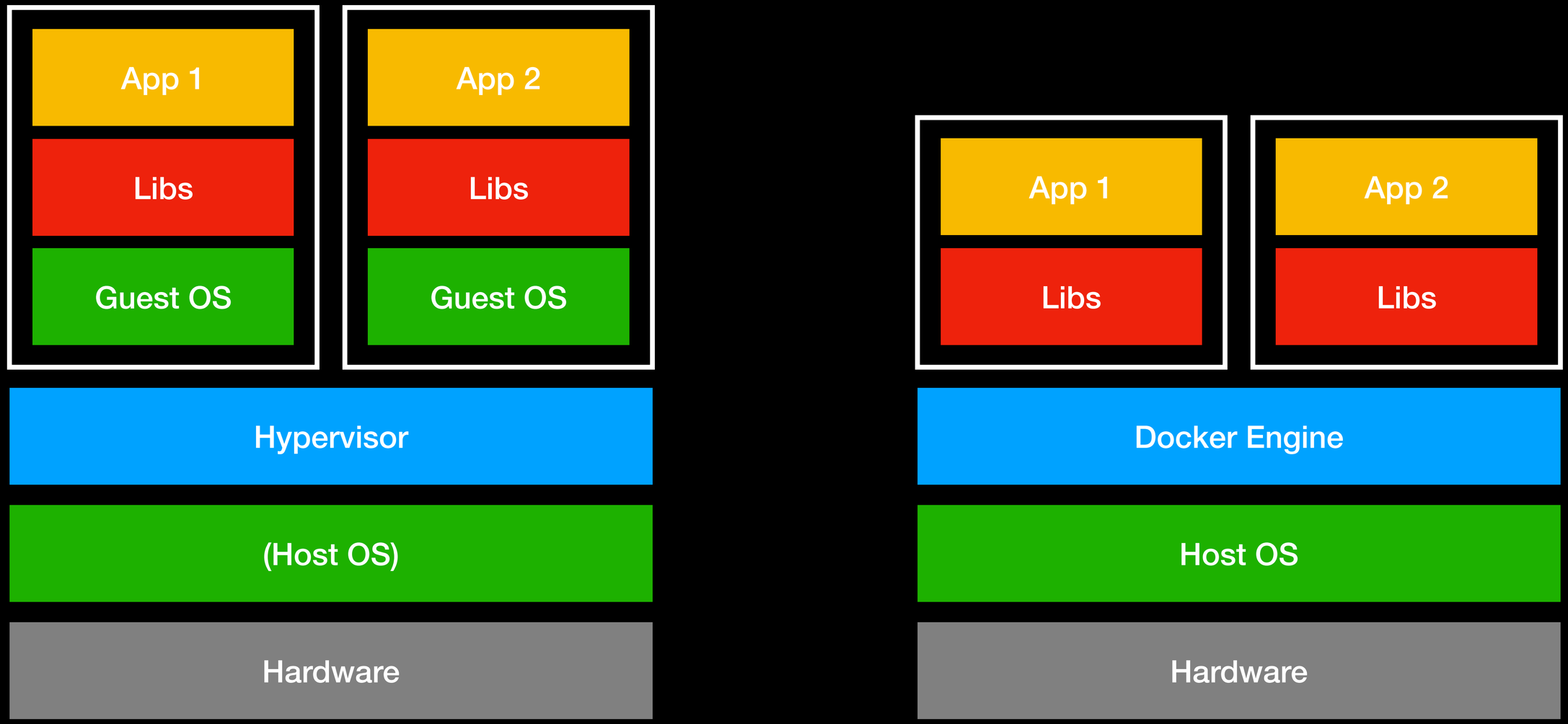


Leichtgewichtige Alternative zu VM



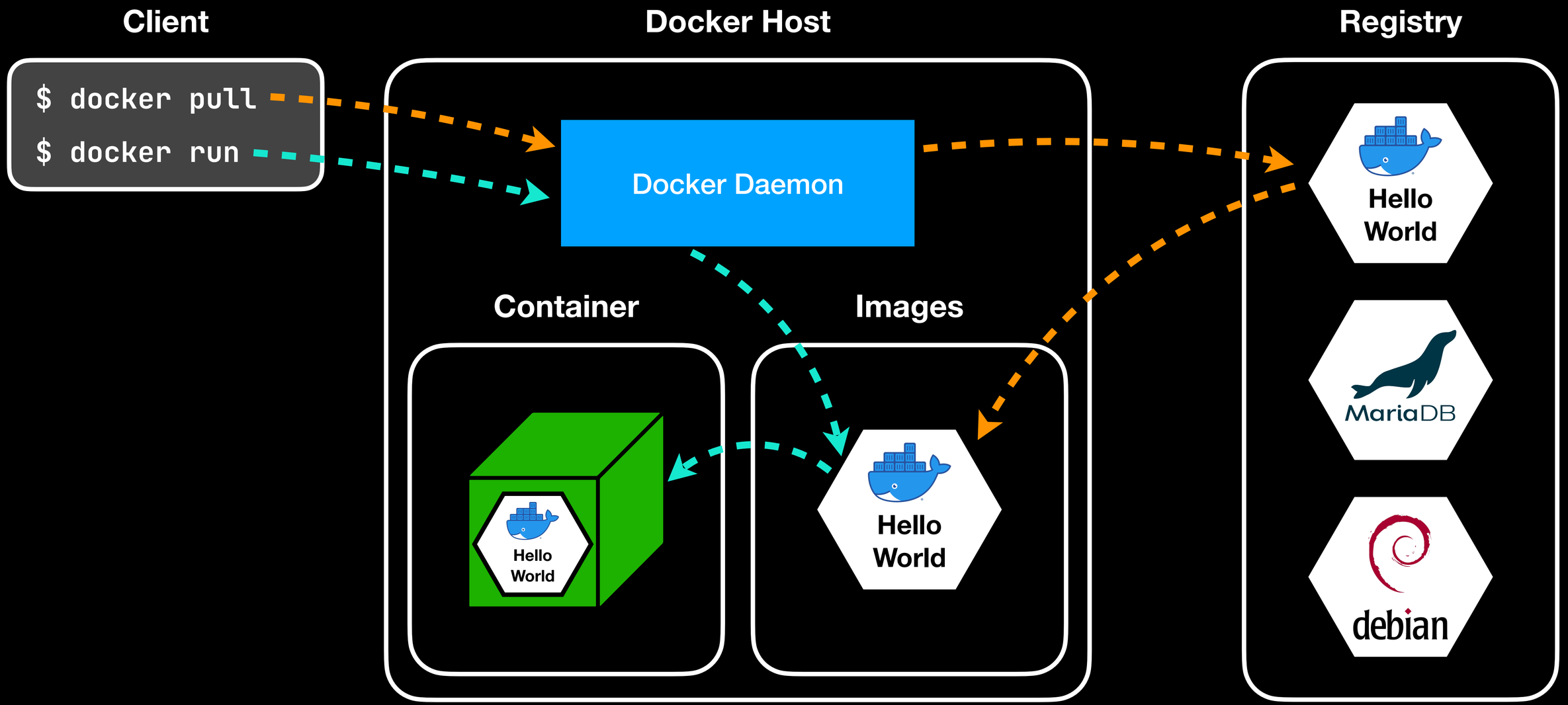
Einfache Verwendung

VM vs Docker


App 1
Libs
Guest OS
App 2
Libs
Guest OS
Hypervisor
(Host OS)
Hardware
App 1
Libs
App 2
Libs
Docker Engine
Host OS
Hardware

Coding
01-hello-world

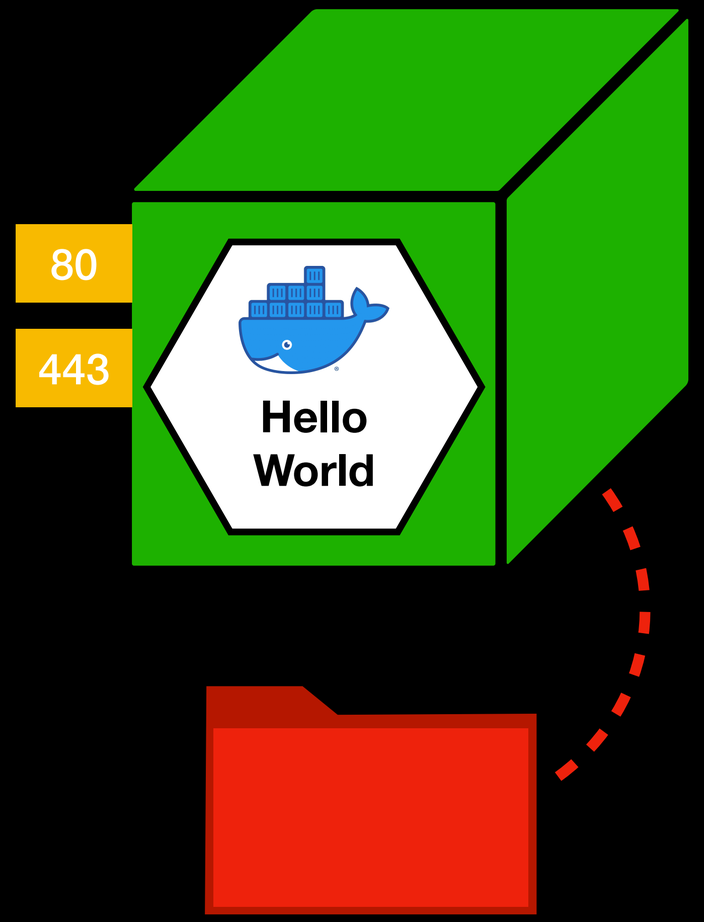
Docker Architektur



https://docs.docker.com/engine/docker-overview

Container

- aus Image erzeugt
- zustandslos → jederzeit neu erstellen
- Port Mappings
- Volume Mappings
- Konfiguration über Umgebungsvariablen



Docker CLI Syntax

```
docker [command]                alte Syntax

docker [object] [command]       neue Syntax
```

Coding
02-container

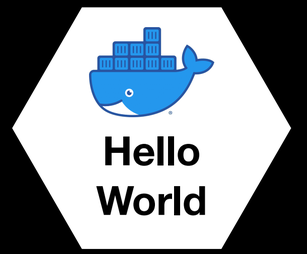
Container

```
$ docker container run          Container erzeugen und starten        docker run
                    -it           interaktiv
                    -d            im Hintergrund
                    -p            Port Mapping
                    -v            Volume Mapping
                    -e            Umgebungsvariable
                ls              Container auflisten                   docker ps
                    -a            inkl. gestoppter Container
                logs            Logs (Standardausgabe) anzeigen       docker logs
                    -f            Log folgen
                exec            Command im Container ausführen        docker exec
                    -it           interaktiv
                start           Container starten                     docker start
                stop            Container stoppen                     docker stop
                rm              Container löschen                     docker rm
                    -f            Laufenden Container löschen
```

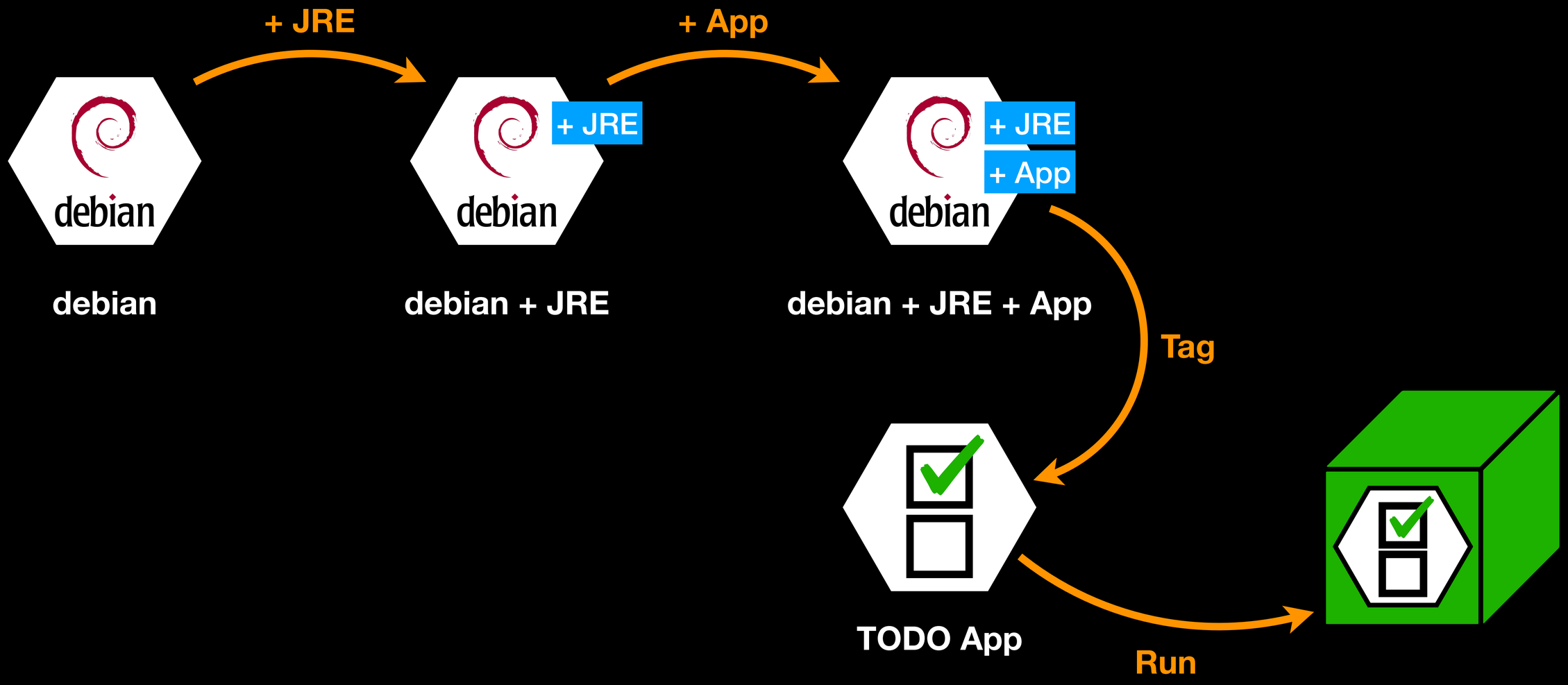

https://docs.docker.com/engine/reference/commandline/container

Image

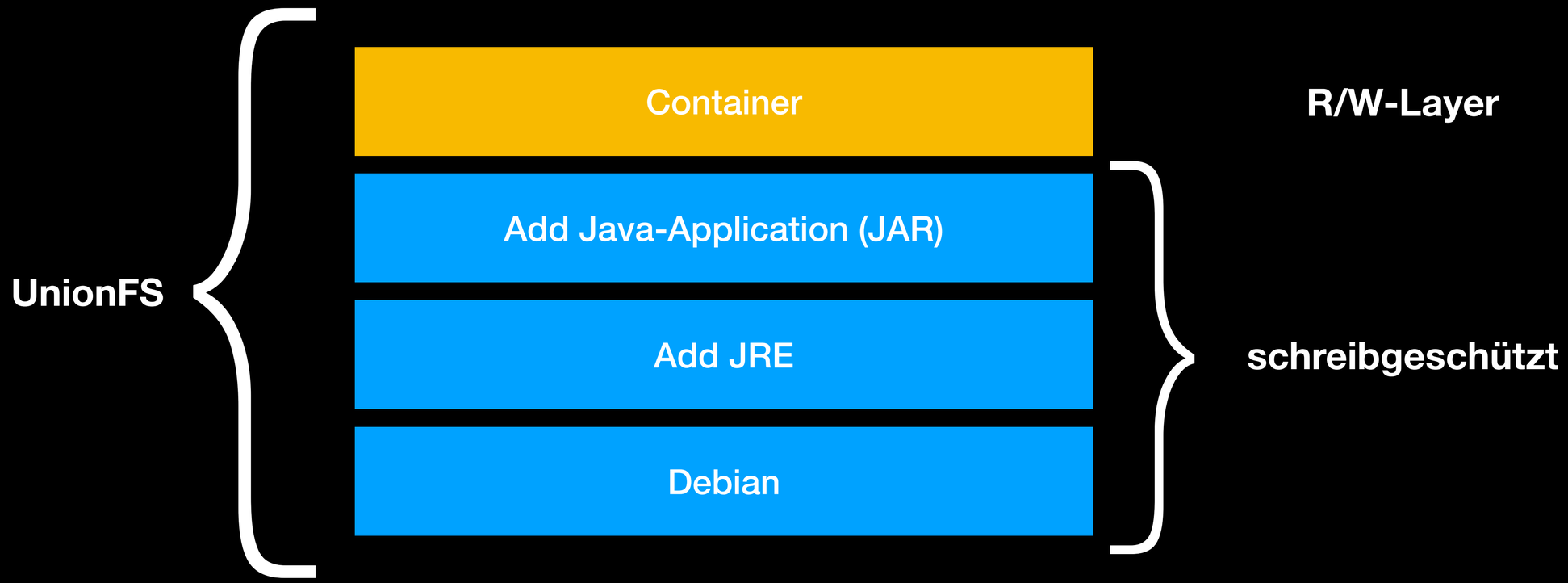
- Vorlage für Container
- schreibgeschützt
- Images können aufeinander aufbauen



Image


+ JRE
+ App
debian
+ JRE
+ JRE
+ App
debian
debian + JRE
debian + JRE + App
Tag
TODO App
Run

Layer


UnionFS
Container
Add Java-Application (JAR)
Add JRE
Debian
R/W-Layer
schreibgeschützt

Layer

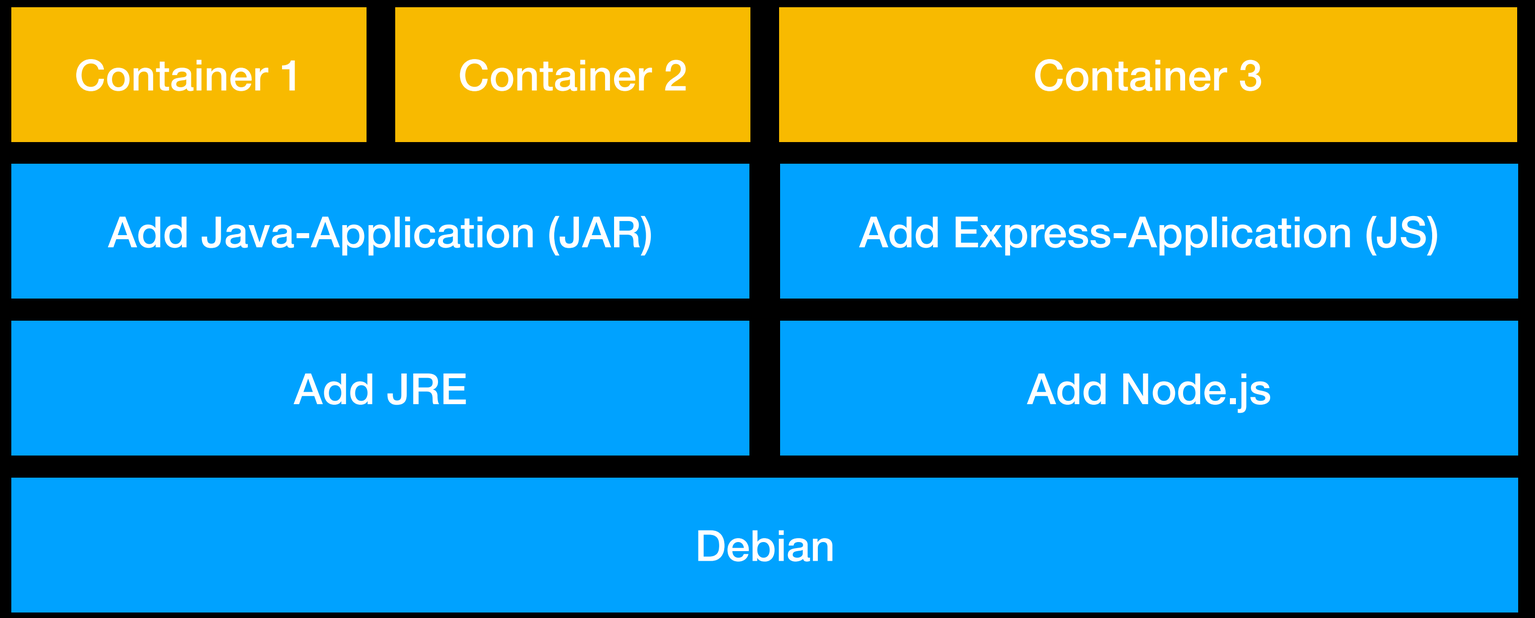
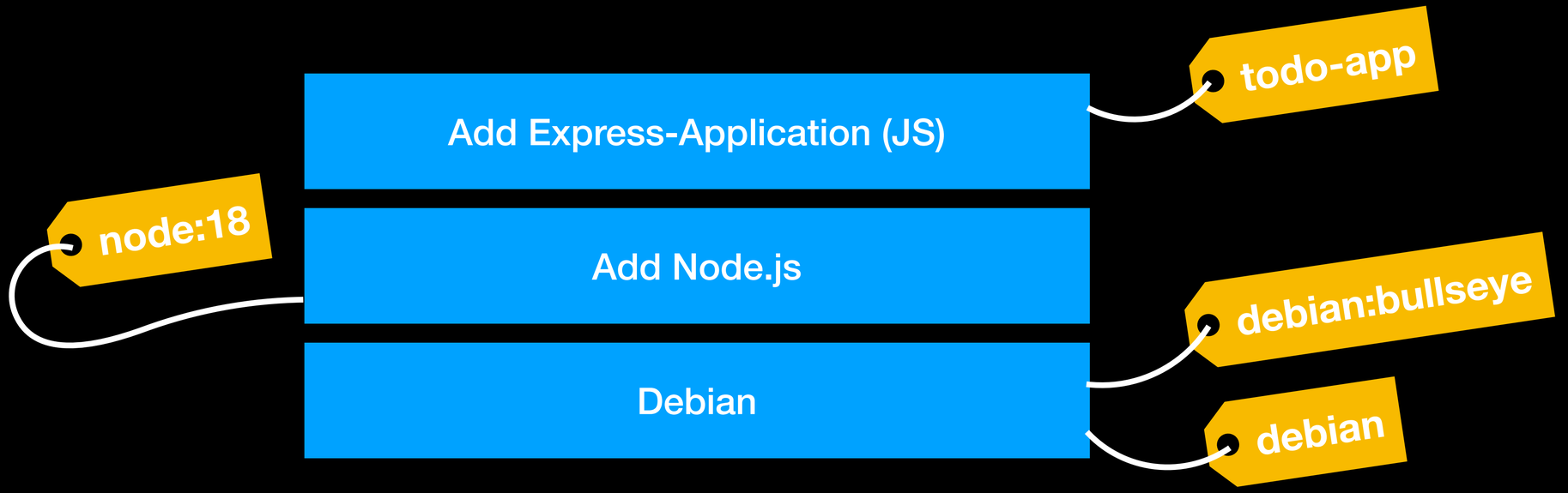

Container 1
Container 2
Container 3
Add Java-Application (JAR)
Add Express-Application (JS)
Add JRE
Add Node.js
Debian

Image Name

- Format: registry / path / to / image / … : tag
  - debian:10
  - my-registry.com/apps/demo:v1.2.3
- Default registry: Docker Hub
- Default tag: latest

Image Name


Add Express-Application (JS)
Add Node.js
Debian
todo-app
node:18
debian:bullseye
debian

Dockerfile

```
FROM <image>

COPY <src>… <dest> (und ADD)

RUN executable param1 param2 ...
RUN ["executable", "param1", "param2", ...]

WORKDIR <path>

CMD executable param1 param2 ...
CMD ["executable", "param1", "param2", ...]

ENV <key>=<value>

EXPOSE <port>
```


https://docs.docker.com/engine/reference/builder

Coding
03-dockerfile

Best Practices

- „ephemeral“ Design
- eine Aufgabe pro Container
- Anzahl an Layers reduzieren
- Layers sinnvoll ordnen
- keine unnötigen Sachen installieren
- `.dockerignore` nutzen
- `apt-get update && apt-get install` in einer Zeile
- Build Cache nutzen
- Multi-Stage Build
- Formatierung

https://docs.docker.com/develop/develop-images/dockerfile_best-practices

Build Cache

- Layers werden gecached und (wenn möglich) wiederverwendet
- Cache Überprüfung
  - `ADD/COPY`: Prüfsumme über Dateiinhalte
  - `RUN`: Command selbst wird abgeglichen
  - Bei Änderungen: Cache-Invalidierung
- Cache-Invalidierung: Alle nachfolgenden Commands nutzen keinen Cache

Image

`$ docker image build`	Image erzeugen	`docker build`
`-t`	Imagename	
`ls`	Images auflisten	`docker images`
`pull`	Image herunterladen	`docker pull`
`push`	Image hochladen	`docker push`
`tag`	Name vergeben	`docker tag`
`rm`	Image löschen	`docker rmi`

https://docs.docker.com/engine/reference/commandline/container

Multi-Stage Build

- Problem mit einfachen Dockerfiles
  - Image-Größe reduzieren umständlich
  - Source Code und Build Tools nach Build nicht benötigt
- Lösung: Multi-Stage Dockerfile
  - Kompilieren erfolgt in einem Container (Stufe 1)
  - Ausführen erfolgt in einem anderen (Stufe 2)

Coding
04-multi-stage-build

Docker Compose

- Problem
  - Lange Run-Befehle
  - Container-Verbunde manuell aufsetzen ist aufwendig
  - Netzwerk zwischen Containern manuell aufsetzen
- Lösung
  - Shell-Skripte
  - Besser: `docker-compose.yml` und `$ docker compose up`

docker-compose.yml

- Einfache Syntax (YAML)
- deklarative Beschreibung von Services (= Containern)
  - Image
  - Ports
  - Volumes
  - Umgebungsvariablen
  - …
- Standard: Alle Container im selben Netzwerk

Coding
05-docker-compose

Docker Compose

```
$ docker compose up                (Images erzeugen,)
                                   Container (erstellen und) starten
                   -d                im Hintergrund
                 ps                Container auflisten
                    -a               inkl. gestoppter Container
                 logs              Logs (Standardausgabe) anzeigen
                   -f                Log folgen
                 exec              Command im Container ausführen
                   -it               interaktiv
                 start             Einzelne Container starten
                 stop              Einzelne Container starten
                 down              Container stoppen und löschen
                   -v                dazugehörige Volumes löschen
                 build             Nur Images erzeugen
```


https://docs.docker.com/engine/reference/commandline/container

Tipps, Tricks und Nützliches

- Programme/Server einfach und unkompliziert testen
- „It works on my machine“ → „It works on **any** machine“
- lokale Datenbank zum Entwickeln
- traefik
- GitHub CI
- IDE Support
- Visual Studio Code Dev Containers

Cleanup

- Container löschen
- Images löschen
- Docker Desktop Reset

Coding
06-tipps

… und weiter?

kubernetes

Build it.
Ship it.
docker run -it

Stefan Schöberl
schoeberl.dev

stefanschoeberl

Images

- https://www.docker.com/company/newsroom/media-resources/
- https://mariadb.com/about-us/logos/
- https://www.debian.org/logos/index.de.html
- https://github.com/cncf/artwork/tree/master/projects/kubernetes