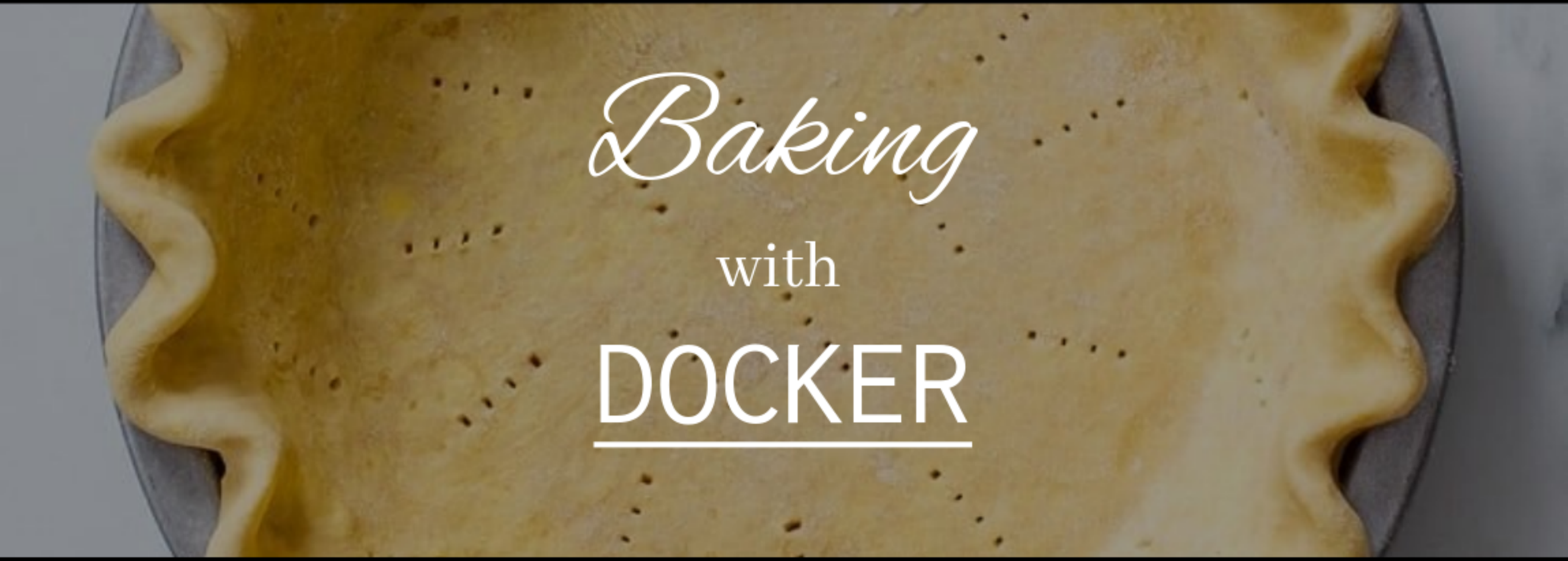
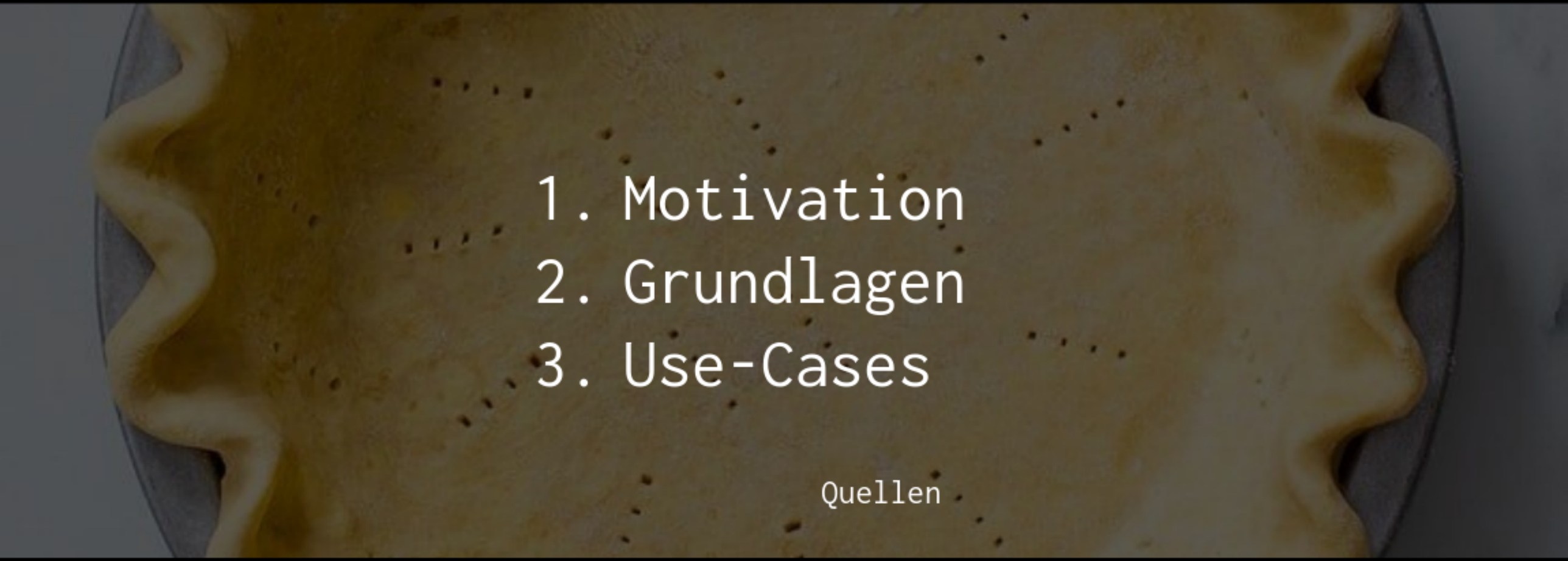


Alex Zimmermann

Themen:

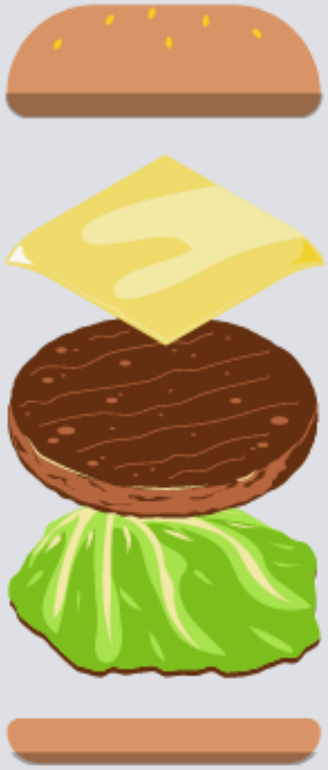


1. Motivation
2. Grundlagen
3. Use-Cases

Quellen

Motivation

Tech-Stack:

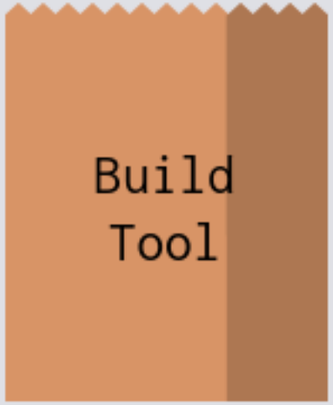


Linux

Apache

MySQL

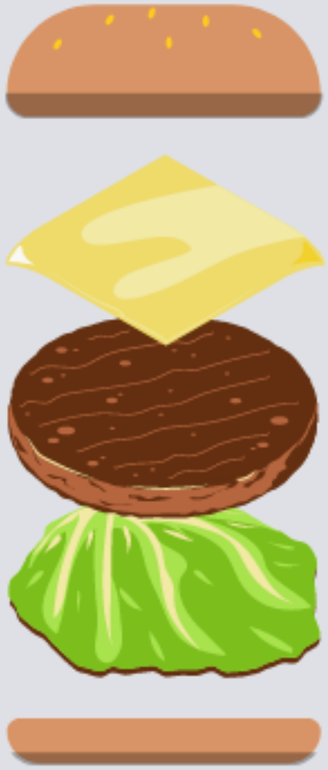
PHP



Webseite

Motivation

Tech-Stack:

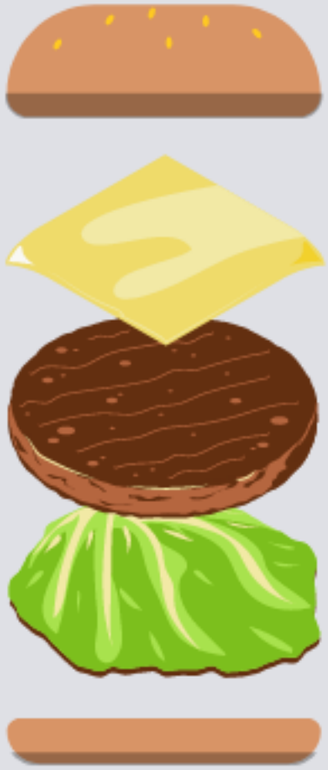


- Linux*
- Apache
  - Version
  - Dependency
    - Dep 1
      - Version(OS)
      - Deps
      - (Config)
    - Dep 2
      - Version(OS)
      - Deps
      - (Config)
    - Dep 3
      - Version(OS)
      - Deps
      - (Config)
  - Config, Env
- MySQL
- PHP

Handelt i.d.R. das Build-Tool* (npm,maven,pip,...)

Motivation

Tech-Stack:



- Linux
- Apache
  - Version
  - Dependency
  - Config, Env
- MySQL
- PHP

Aufsetzen und Testen dauert 1-2 Mantage

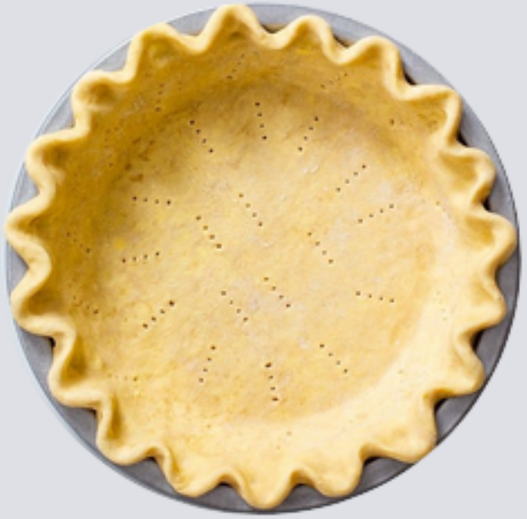
und Kopfschmerzen

Motivation

Wunsch-Vorstellung

Fokus auf Entwicklung und Code

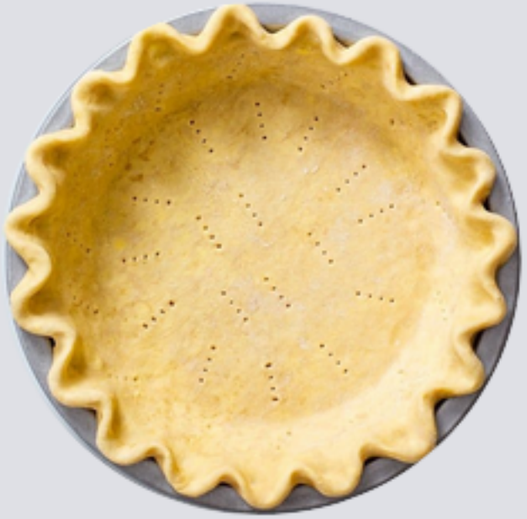
Fertige Umgebung + Code, Assets, etc. = Endprodukt



```
int main(int argc, char** argv)
{
  https://docs.docker.com;
  return 0;
}
```

Motivation

Docker to the rescue



Image

```
int main(int argc, char** argv)
{
  https://docs.docker.com;
  return 0;
}
```

User(Host)-Code

Container

Grundlagen

Image

- OS (Ubuntu, Alpine)
- Libs/Tools (java, python, ...)
- Umgebungsvariablen (JAVA_HOME)
- Dateien/Ordner (auch Code)
- Einstiegspunkt (Run-Befehl)
- definiert durch Dockerfile

Host

- erstellt Images
- startet/verwaltet Container
- ist der Rechner auf dem Docker installiert ist

Container

- die Instanz einer Image
- Port-Mappings
- Volume-Mappings
- auch andere Laufzeitparameter
- Start/Stop Verhalten

"So schuf Host die Container nach seinen Images ..."

- 1.Buch Mose 27

Grundlagen

Zusammenfassend:
Ein Rechner hostet einen Container
nach dem Bauplan eines Image.

How to Image

Alex Zimmermann

Schritt 1: Dockerfile schreiben

```
FROM python:latest



# more to come ...



CMD ["python", "-m", "http.server"]
```

Nutze image_name:tag_name als Basis

Starte dieses Kommando (CMD) bei Ausführung in einem Container: python -m http.server

How to Image

Alex Zimmermann

Schritt 1: Dockerfile schreiben

```
FROM python:latest

COPY entrypoint.py ./
COPY reqs.txt ./

RUN pip install -r reqs.txt



CMD ["python", "entrypoint.py"]
```

Nutze image_name:tag_name als Basis

Kopiere Einstiegspunkt und Metafiles von Host nach Image

Installiere Abhängigkeiten, starte Tests ...

Starte dieses Kommando (CMD) bei Ausführung in einem Container:
python -m http.server

How to Image

Alex Zimmermann

Schritt 2: Image bauen

```
> docker build -t hello_image .
```

```
FROM python:latest


CMD ["python", "-m", "http.server"]
```

```
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM python:latest
latest: Pulling from library/python
8457fd5474e7: Downloading [==================================>                ]  35.99MB/49.58MB
13baa2029dde: Download complete
325c5bf4c2f2: Downloading [==================================>                ]  44.63MB/64.13MB
7e18a660069f: Downloading [=====>                                             ]  22.02MB/211.1MB
98a59f0ffede: Waiting
72c7f17f2221: Waiting
2f40b346325a: Waiting
f3f08e04e337: Waiting
```

How to Image

Alex Zimmermann

Schritt 2: Image bauen

```
> docker build -t hello_image .
```

```
FROM python:latest


CMD ["python", "-m", "http.server"]
```

Fertig!
Dauert beim
ersten Pull gerne
länger

```
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM python:latest
latest: Pulling from library/python
8457fd5474e7: Pull complete
13baa2029dde: Pull complete
325c5bf4c2f2: Pull complete
7e18a660069f: Pull complete
98a59f0ffede: Pull complete
72c7f17f2221: Pull complete
2f40b346325a: Pull complete
f3f08e04e337: Pull complete
Digest: sha256:89f4c413ac0f36072211bced42ff7e8870cf5347c3cde4b84a67b5f87911b9a3
Status: Downloaded newer image for python:latest
 ---> 2acccf902fa3
Step 2/2 : CMD ["python", "-m", "http.server"]
 ---> Running in 62f632abea14
Removing intermediate container 62f632abea14
 ---> e367f6ec2e76
Successfully built e367f6ec2e76
Successfully tagged hello_image:latest
[alex@alex-pc docker-test]$
```

How to Container

Alex Zimmermann

Container starten:

```
> docker run --name=hello_container hello_image
```

```
[alex@alex-pc docker-test]$ sudo docker run --name=hello_container hello_image
```

How to Container

Alex Zimmermann

Container starten:

```
> docker run --name=hello_container hello_image
```

```
[alex@alex-pc docker-test]$ sudo docker run --name=hello_container hello_image
```

Läuft das Ding ?

How to Container

Alex Zimmermann

Status aller laufenden Container prüfen:

```
> docker ps
```

```
[alex@alex-pc ~]$ sudo docker ps
[sudo] Passwort für alex:
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS     NAMES
b623d41d2b81   hello_image   "python -m http.serv…"   25 seconds ago   Up 24 seconds             hello_container
[alex@alex-pc ~]$
```

How to Container

Status aller laufenden Container prüfen:

```
> docker ps
```

```
[alex@alex-pc ~]$ sudo docker ps
[sudo] Passwort für alex:
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS     NAMES
b623d41d2b81   hello_image   "python -m http.serv…"   25 seconds ago   Up 24 seconds             hello_container
[alex@alex-pc ~]$
```

Aber lokaler Test sagt Nein:

```
[alex@alex-pc ~]$ curl localhost:8000
curl: (7) Failed to connect to localhost port 8000 after 0 ms: Couldn't connect to server
[alex@alex-pc ~]$
```

How to Container

Fehler ?

How to Container

Fehler: Information über Ports fehlt komplett.

```
FROM python:latest

EXPOSE 8000

CMD ["python", "-m", "http.server"]
```

```
> docker run --name=hello_container -p 8000:8000 hello_image
```

```
[alex@alex-pc ~]$ sudo docker ps
[sudo] Passwort für alex:
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS   NAMES
b623d41d2b81   hello_image    "python -m http.serv…"   25 seconds ago   Up 24 seconds   ???     hello_container
[alex@alex-pc ~]$
```

How to Container

Nochmal prüfen:

```
[alex@alex-pc ~]$ curl localhost:8000
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
```

OK

Use-Cases

Alex Zimmermann

Was kann man damit noch so anstellen ?

Use-Cases

Was kann man damit noch so anstellen ?

Automatische Tests:

```
FROM python:3

# Code kopieren etc.

CMD ["python", "run_tests.py"]
```

```
FROM node:lts

# Code kopieren etc.

CMD ["npm", "test"]
```

```
FROM java:11

# Code kopieren etc.

CMD ["mvn", "test"]
```

Use-Cases

Was kann man damit noch so anstellen ?

Automatische Builds:

```
FROM node:lts

# Code kopieren etc.

CMD ["npm", "build"]
```

Bräuchte hier ein Volumen, sonst eher nutzlos.

Der docker-run Befehl hat einen fehlerfreien Rückgabewert, falls Tests erfolgreich. Der so entstandene Build kann z.B. in eine Repo ge-published werden.

Use-Cases

Was kann man damit noch so anstellen ?

Ein bestimmtes Tool nicht installiert ?

-> Kein Problem. Einfach in Docker "laden":

```
> docker run -it php:latest php -a
```

Startet ein PHP REPL

```
> docker run -d bitnami/moodle
```

Startet eine lokale Moodle Instanz

Nicht Zuhause nachmachen!

Use-Cases

Was kann man damit noch so anstellen ?

DevOps leicht gemacht:

```
> docker run --restart=always bitnami/moodle
```

Startet den Container im letzten Zustand auch nach einem Rechner-Neustart wieder. (min. mit systemd)

```
> docker exec -it my_moodle bash
```

Startet eine Bash-Shell im laufenden Container my_moodle.

Use-Cases

Was kann man damit noch so anstellen ?

"Live" Coding/Debugging:

```
> docker run -v my_code:/path/to/app bitnami/moodle
```

Schreibe Code lokal und die Änderungen
werden im Container reflektiert.

Use-Cases

Was kann man damit noch so anstellen ?

Kaputtes Linux wieder reparieren, weil man zuviel Kaffee konsumiert hat und zulange wach war und deshalb "aus Versehen" einen wichtigen Ordner des OS per sudo zerschossen hat.

```
> docker run -it archlinux:base bash
```

Mit chroot kann man die kaputte Ordnerstuktur wieder herstellen.

Use-Cases

Aber ich will jetzt trotzdem nicht
alles "vom Scratch" ewig konfigurieren
und testen bis es endlich läuft ?!

Use-Cases

```
Aber ich will jetzt trotzdem nicht
alles "vom Scratch" ewig konfigurieren
und testen bis es endlich läuft ?!
```

https://hub.docker.com ist Dein Freund.

- viele Kombinationen von Software bereits vorhanden
- Images werden regelmäßig gewartet
- Bedienungsanweisungen inklusive
- Notfalls: von (FROM) einer halbwegs guten Image ableiten und ändern

Use-Cases

https://hub.docker.com ist Dein Freund.

Bonus Tipp:

"alpine" Images
nutzen

Bonus

Warum kann man so viele Container gleichzeitig laufen lassen ?

Bonus

Warum kann man so viele Container gleichzeitig laufen lassen ?

- Ausführung der Container auf Prozess-Ebene
- Isolation per cgroups und namespaces
- erlaubt Multithreading u.a. OS Parallelisierungen (vgl. green threads, coroutines)
- nicht das gesamte Basis-OS wird simuliert (HOST kernel wird genutzt)
- OS Abhängigkeiten können minimiert werden (vgl. alpine)