Containerisierung
at Swiss Post IT

DIE POST

What are we doing today?
Table of Content

- Introducing round
- Middleware Engineering
- What is a Container?
- Docker Container Introduction
- Docker installation
- Docker Examples
  - Hello World
  - Webserver
  - WikiJS
  - Filebrowser
  - Gitea
- Kubernetes Demo

Introducing Round

Who are we? Who are you?

Who we are

Patrick Bühlmann
Middleware Engineering

Work area:

- Kubernetes / OpenShift Engineering
- Docker, Cloud, Automation
- DevOps CI / CD

Telefon +41 76 342 82 59
Fax -
E-Mail patrick.buehlmann.1@post.ch

Nathanael Liechti
Middleware Engineering - Lernender

Work area:

- Kubernetes Engineering
- Go Coding (Microservices and more)

Telefon +41 76 416 81 33
Fax -
E-Mail nathanael.liechti@post.ch

Who are you?
A short introduction

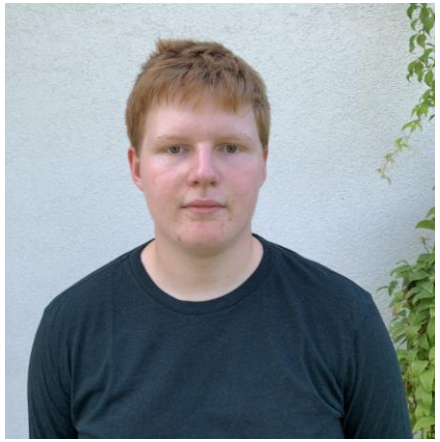
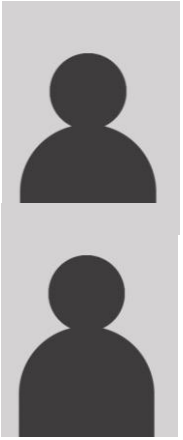
- What is your name?
- Do you have experience with Docker, Kubernetes or Container in general?
- How motivated are you from a scale of 0 (terrible) to 10 (very good)?

Middleware Engineering – I253

Our team









Middleware Engineering
What is Engineering?

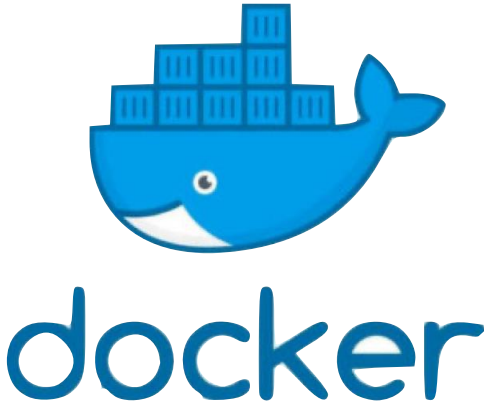
- Systematic design of computer science systems
- Create and implement concepts / products

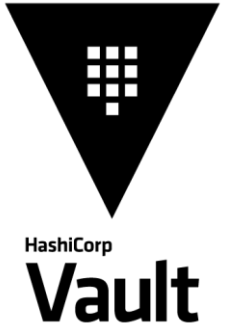
ICT-System-Ingenieur:

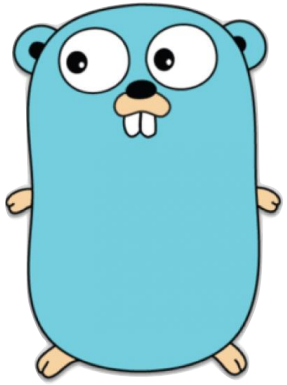
Planning, procurement, commissioning, testing and acceptance of platforms (hardware, software, networks, including cloud environments) for the operation of ICT systems; definition of operational requirements

https://www.berufe-der-ict.ch/berufe/entwicklung/ict-system-ingenieur

Middleware Engineering
Engineering Tools







Go

Middleware Engineering
What does an Engineer do in his daily business?

- Scripts, Automate everything (seriously, **everything**!)
- Terminal, Shell
- Work with Linux all day long
- Searching and developing new solutions
- Try somthing new
- Questioning the existing
- Interest in system / network / cloud

Docker Container
Installation of Docker

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-debian-10

Docker Container
Installation of Docker – Smoke-Tests

```
[user@hostname /]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

```
[user@hostname /]# docker version
Client: Docker Engine - Community
 Version:           20.10.12
 API version:       1.41
 Go version:        go1.16.12
 Git commit:        e91ed57
 Built:             Mon Dec 13 11:45:48 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.12
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.12
  Git commit:       459d0df
  Built:            Mon Dec 13 11:43:56 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.12
  GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:          1.0.2
  GitCommit:        v1.0.2-0-g52b36a2
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Container

What is a container?

Container

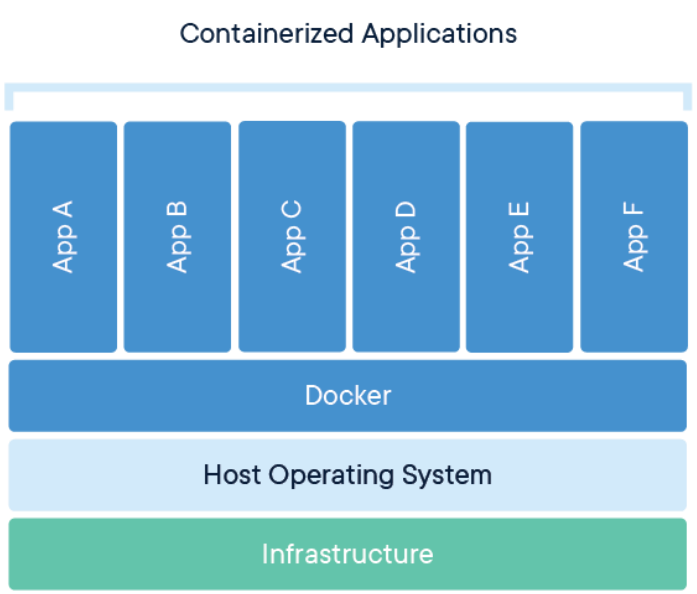
What is a Container?

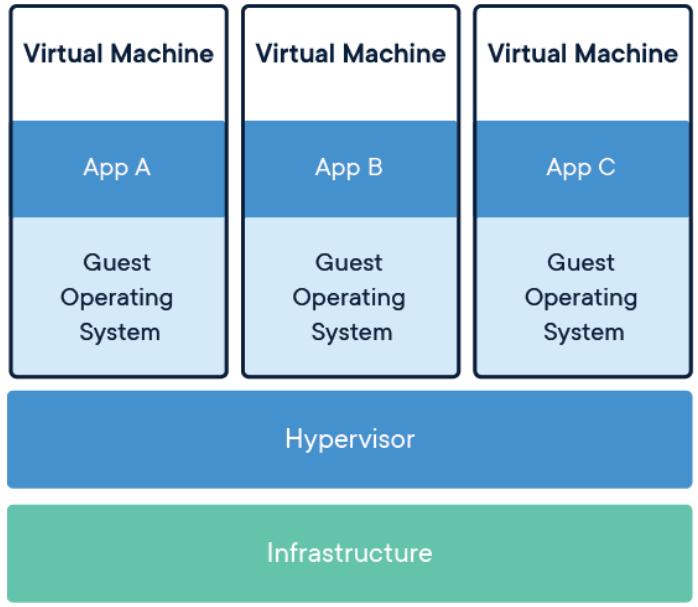
"Containers are a form of operating system virtualization. A single container might be used to run anything from a small microservice or software process to a larger application. Inside a container are all the necessary executables, binary code, libraries, and configuration files. Compared to server or machine virtualization approaches, however, containers do not contain operating system images. This makes them more lightweight and portable, with significantly less overhead. In larger application deployments, multiple containers may be deployed as one or more container clusters.

Such clusters might be managed by a container orchestrator such as Kubernetes."

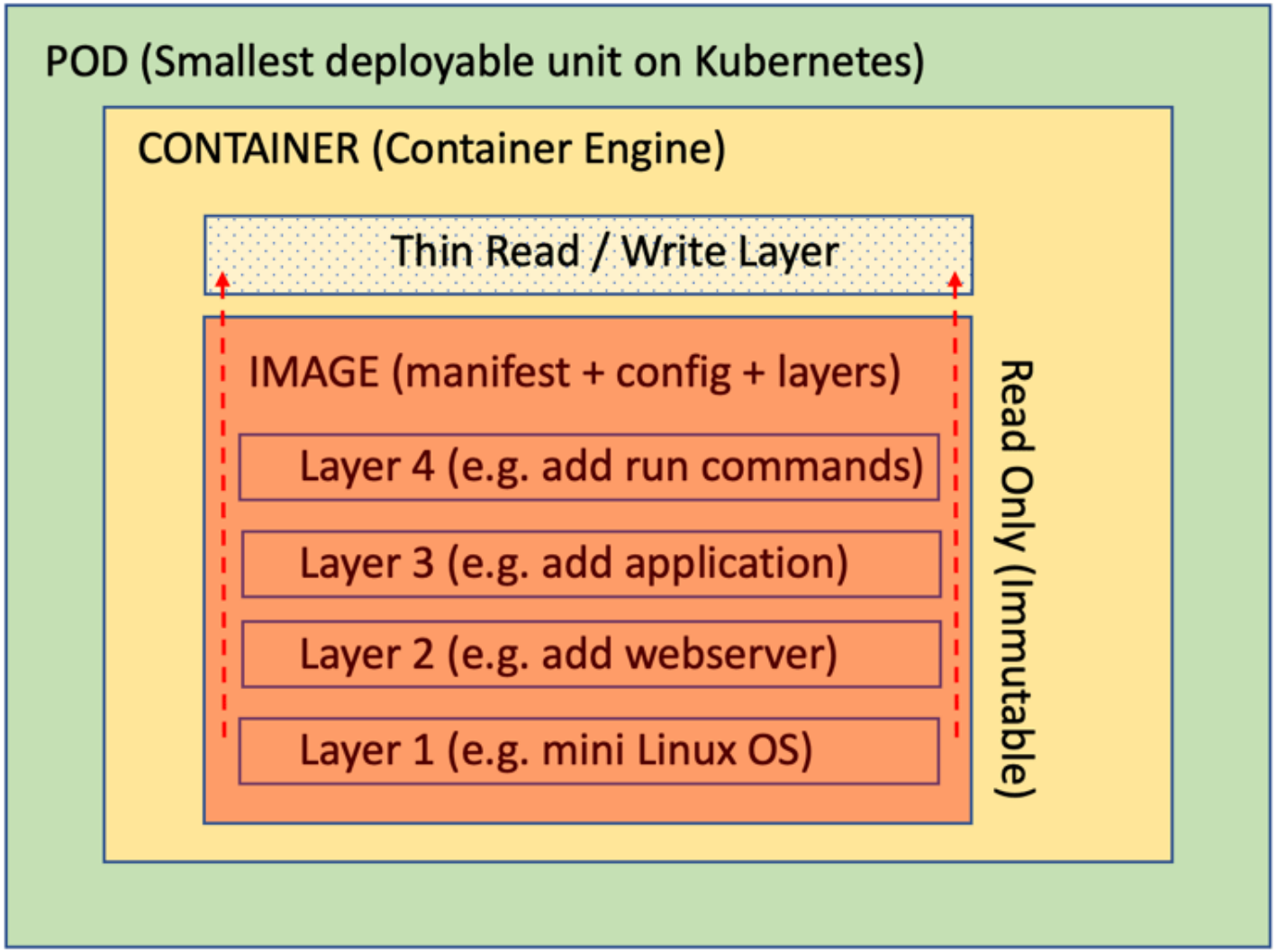
Container
Comparison between Container and Virtual Machines

- Share kernel and isolate the application process
- Extremly portable
- Very lightweight
- One process in a container
- No data in a container (use volumes)

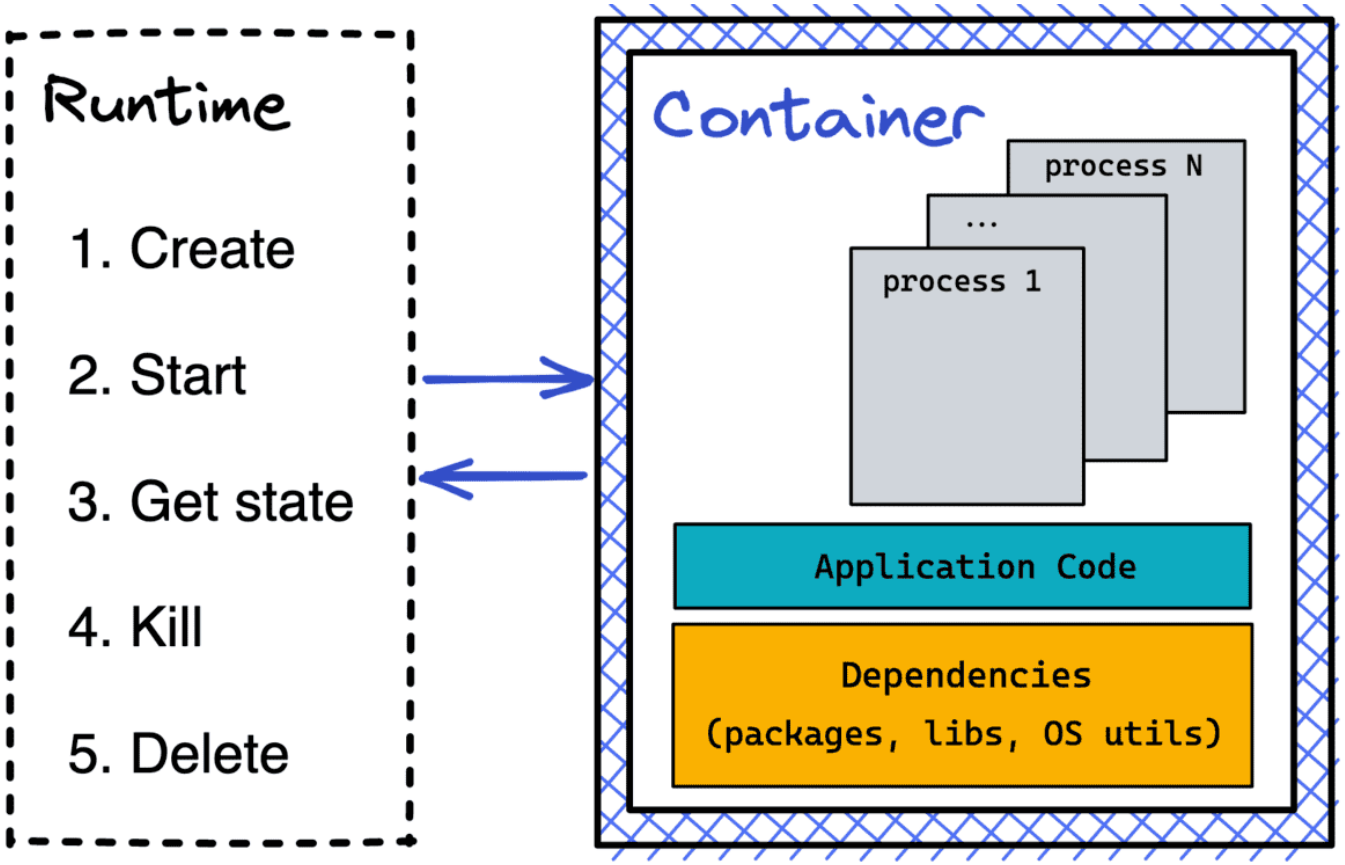




Container
Inside of a Container



Container
Interact with a Container



Containers
work the same way on

laptop server cloud

with

Docker/Kubernetes/Podman
etc.

Container
Benefits of containers

- Less overhead
- Increased portability
- More consistent operation
- Greater efficiency
- Easier application development (less dependencies)
- Much faster deployment
- Save costs

Docker

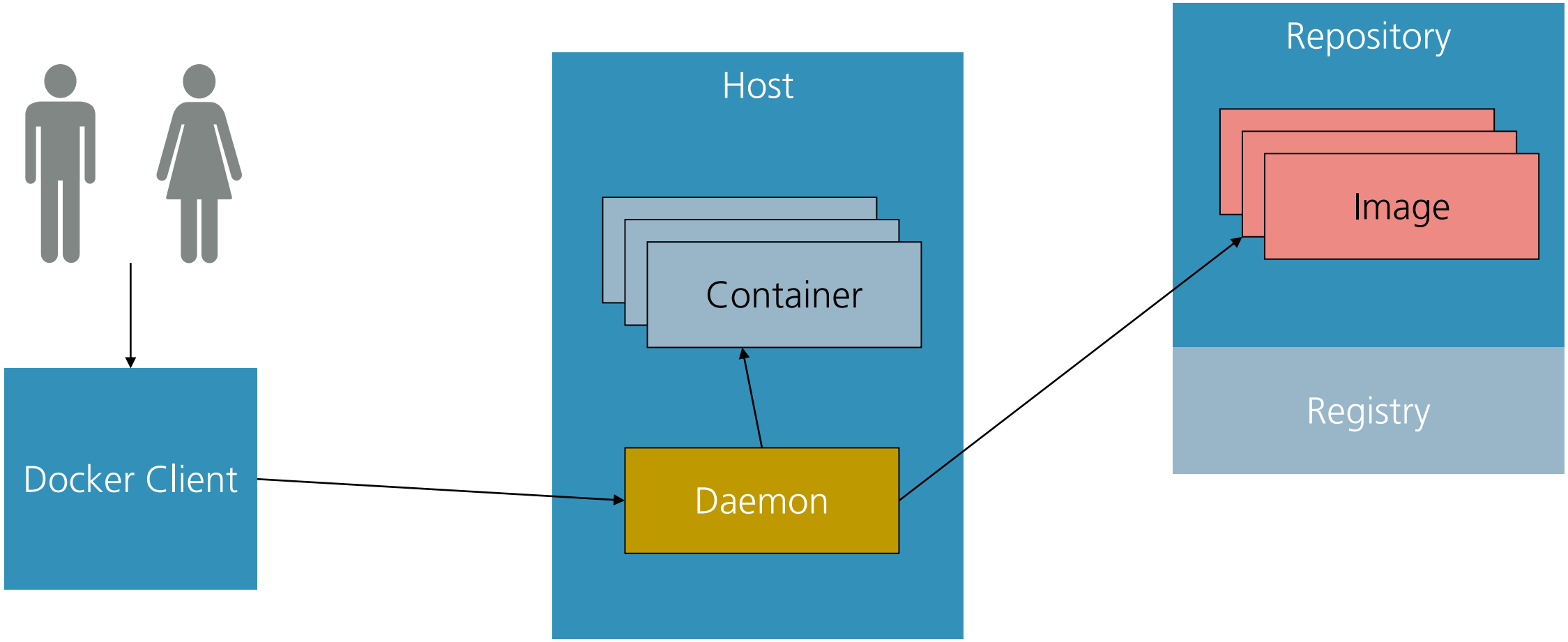
Create, deploy and run Containers

Docker
What is Docker?

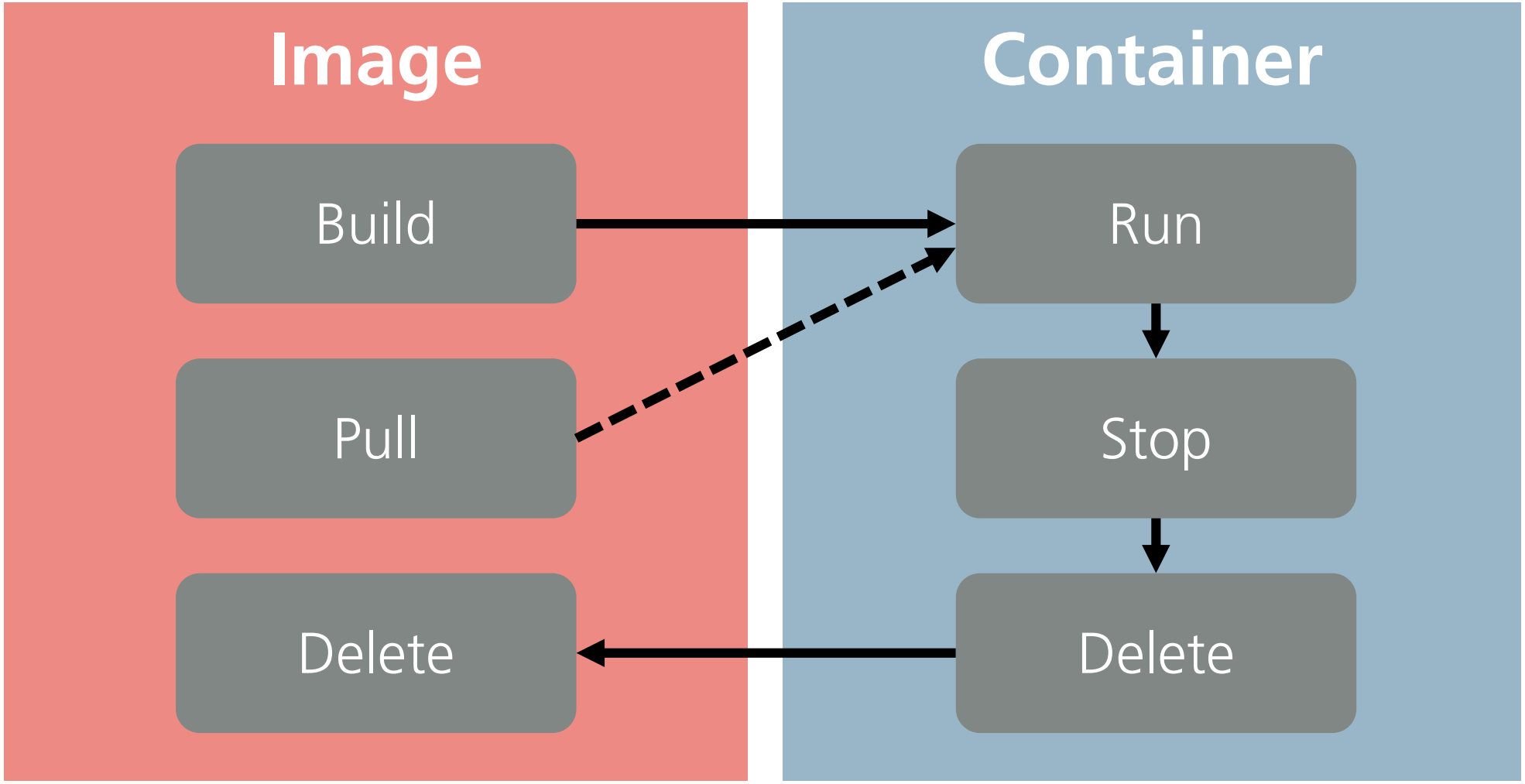
Docker is a software platform designed to make it easier to create, deploy, and run applications by using containers.
It allows developers to package up an application with all the parts it needs in a container, and then ship it out as one package.

Build, share and run any application, anywhere

Docker Container
Docker Mechanism



Docker Container
Workflow



Docker Container
Docker Engine Commands

Image

- build
- pull
- tag
- Rmi

Container

- logs
- inspect
- ps
- Stop
- kill
- rm

- run
- start

Docker Container
Installation of Docker

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-debian-10

Docker Container
Installation of Docker – Smoke-Tests

```
[user@hostname /]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

```
[user@hostname /]# docker version
Client: Docker Engine - Community
 Version:           20.10.12
 API version:       1.41
 Go version:        go1.16.12
 Git commit:        e91ed57
 Built:             Mon Dec 13 11:45:48 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.12
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.12
  Git commit:       459d0df
  Built:            Mon Dec 13 11:43:56 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.12
  GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:          1.0.2
  GitCommit:        v1.0.2-0-g52b36a2
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Docker Container
Docker Runtime environment variables

- **Environment Variables**
  docker run –it \
    -e GREET=HELLO \
    --rm centos

```
[user@hostname /]# env | grep GREET
GREET=HELLO
[user@hostname /]# echo $GREET
HELLO
```

Docker Container
Docker Runtime networking

- **Port translation**
  docker run … -p 10022:22 …

```
"PortBinding": {
    "22/tcp": [
        {
            "HostIp": "",
            "HostPort": "10022"
        }
    ],
    "3000/tcp": [
        {
            "HostIp": "",
            "HostPort": "10080"
        }
    ]
},
```

Docker Container
Docker Runtime volumes

- **Mount a directory from the host into the container**
  docker run … -v /data/db:/var/lib/db …

Docker Container

The dockerfile is the truth

Dockerfile → Image → Container

- Instruction-bases image build
  - FROM, ADD, RUN, COPY, CMD, USER
- Cached layers
- Dockerfile Cheat Sheet →

```
FROM alpine:latest
LABEL maintainer="patrick.buehlmann.1@post.ch"
WORKDIR /application/run
RUN apk add --no-cache bash
COPY run_my_application.sh .
RUN chmod 751 run_my_application.sh
CMD "/hello_world_bash_script.sh"
```

DOCKER EXERCISE «Pre Hello World»

Docker Exercise
Pre Hello World

1. docker run hello-world
2. docker ps –a
3. docker images

DOCKER EXERCISE «Hello World»

Docker Exercise
Hello World

1. Write a Dockerfile that echos «Hello World!» when the container starts
   1. Choose the right, small base image (alpine, centos)
   2. Create a bash script to echo «Hello World!»
2. Build and run the container

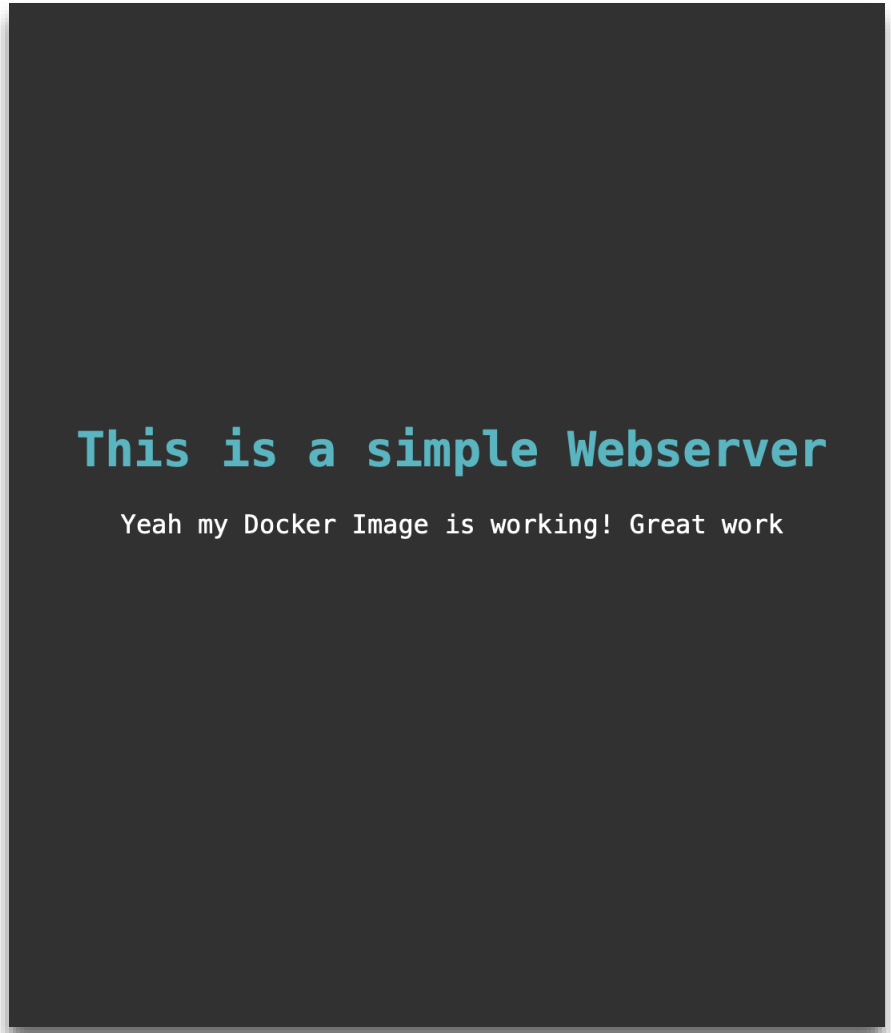
- Is there a faster way?

DOCKER EXERCISE «Webserver»

Docker Exercise

Webserver

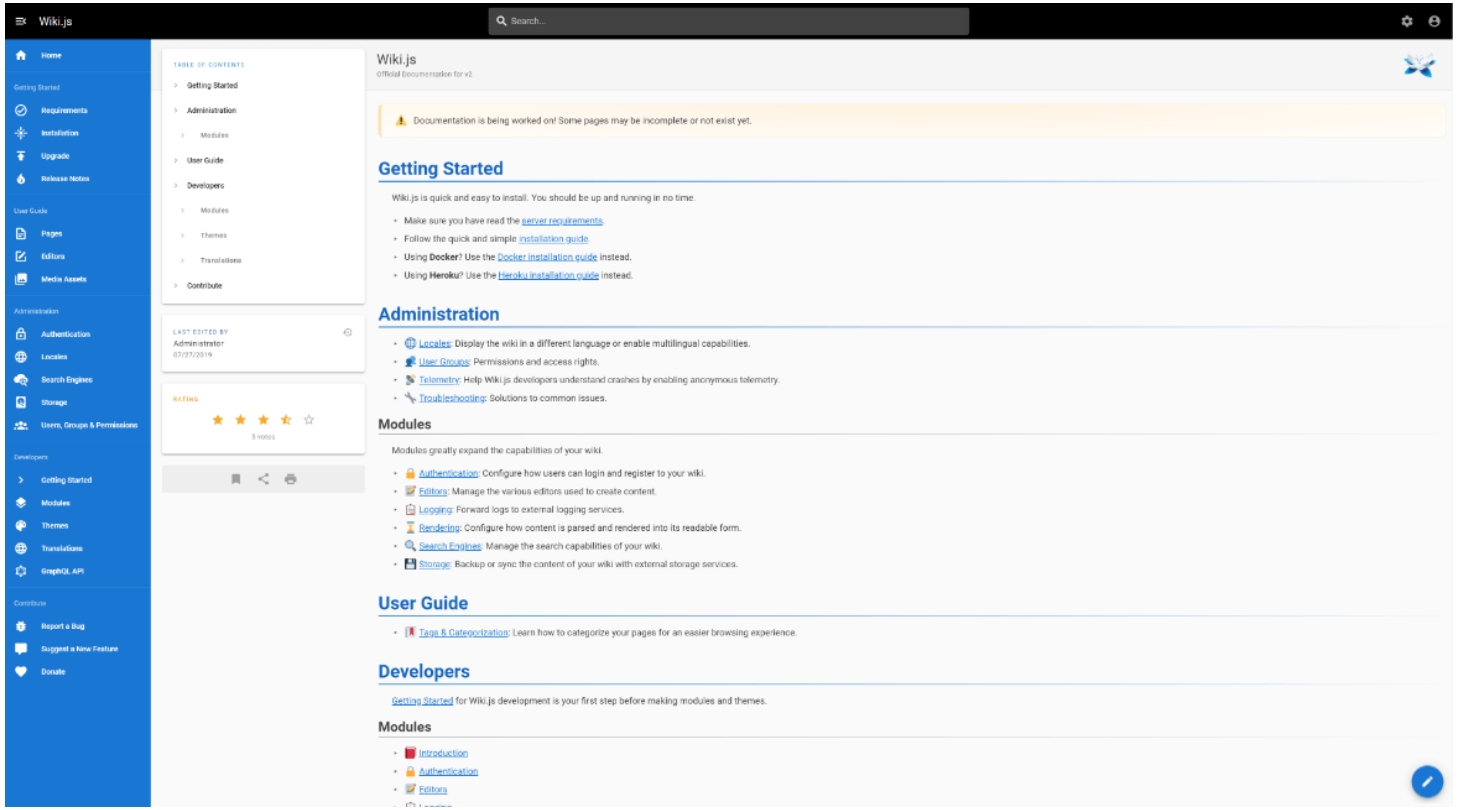
1. Write a Dockerfile for a Webserver showing your custom little HTML Page
   1. Choose the right, small base image (alpine, centos)
   2. Install an apache http server
   3. Create an index.html file and paste it to the web folder of the apache server
2. Build and run the container in the detached mode (background)
3. Start the images«helloworld» and «webserver» with a docker-compose file



DOCKER EXERCISE «WikiJS»

Docker Exercise
WikiJS

1. Run a wikijs container
   1. ENV=DB_TYPE=sqlite
   2. ENV=DB_FILEPATH=/wiki/wiki.sqlite
   3. Mount wiki.sqlite as File
   4. Expose port 3000
2. Finish the setup of wikijs on http://localhost:3000
3. Write your first wiki-page!



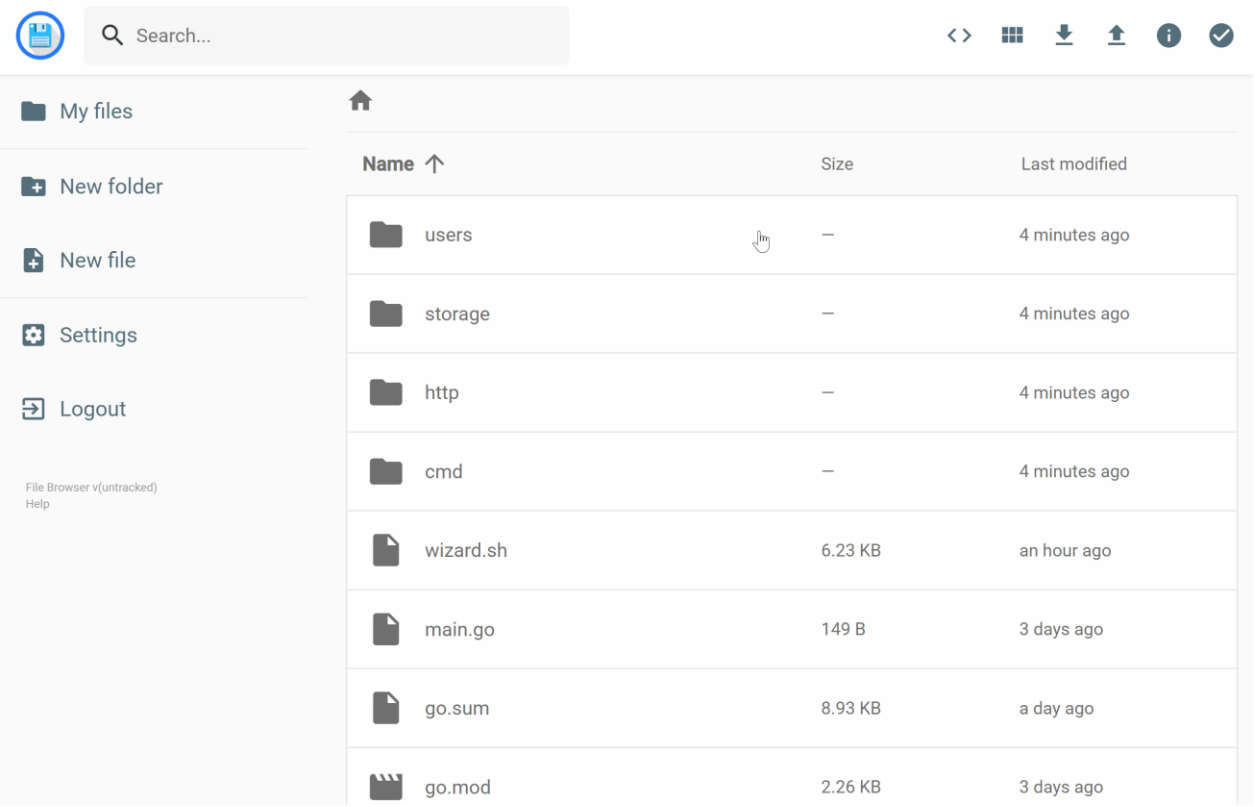
https://docs.requarks.io/install/docker

DOCKER EXERCISE «Filebrowser»

Docker Exercise

Explanation of the exercise

1. Create a docker volume «filebrowser»
2. Run a filebrowser container:
   1. Expose port 80 as 8080
   2. Mount database.db in /database.db as File
   3. Mount docker volume in /src



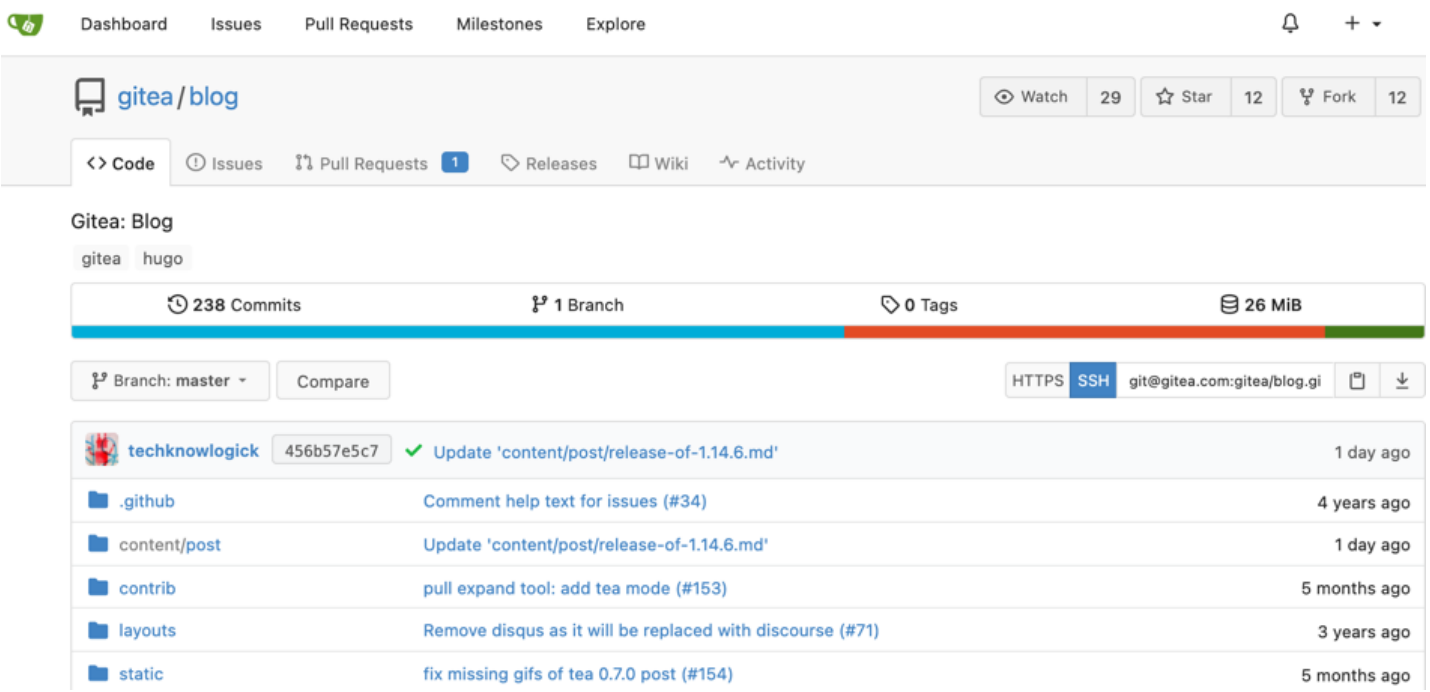
https://filebrowser.org/installation#docker

DOCKER EXERCISE «Gitea»

Docker Exercise

Gitea – self hosted Git

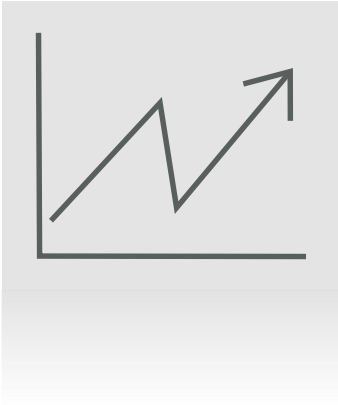
1. Write a docker-compose file
2. docker-compose up -d
3. Finish the setup on http://localhost:3000 (don't forget to create the admin user!)
4. Create and clone a repository
5. What happens when you stop the container?



https://docs.gitea.io/en-us/install-with-docker/

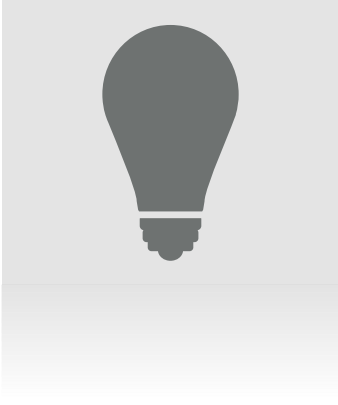
Kubernetes @ DIE POST

Kubernetes@Post
Key Figures



Key figures

- Running Container: 3’800
- Used Container Nodes: 300



Application purpose

- PostAuto Backend
- Chatbot and Livechat
- Postshop Backend
- Automation and Logging Tools

Kubernetes
Containerisation with K8S

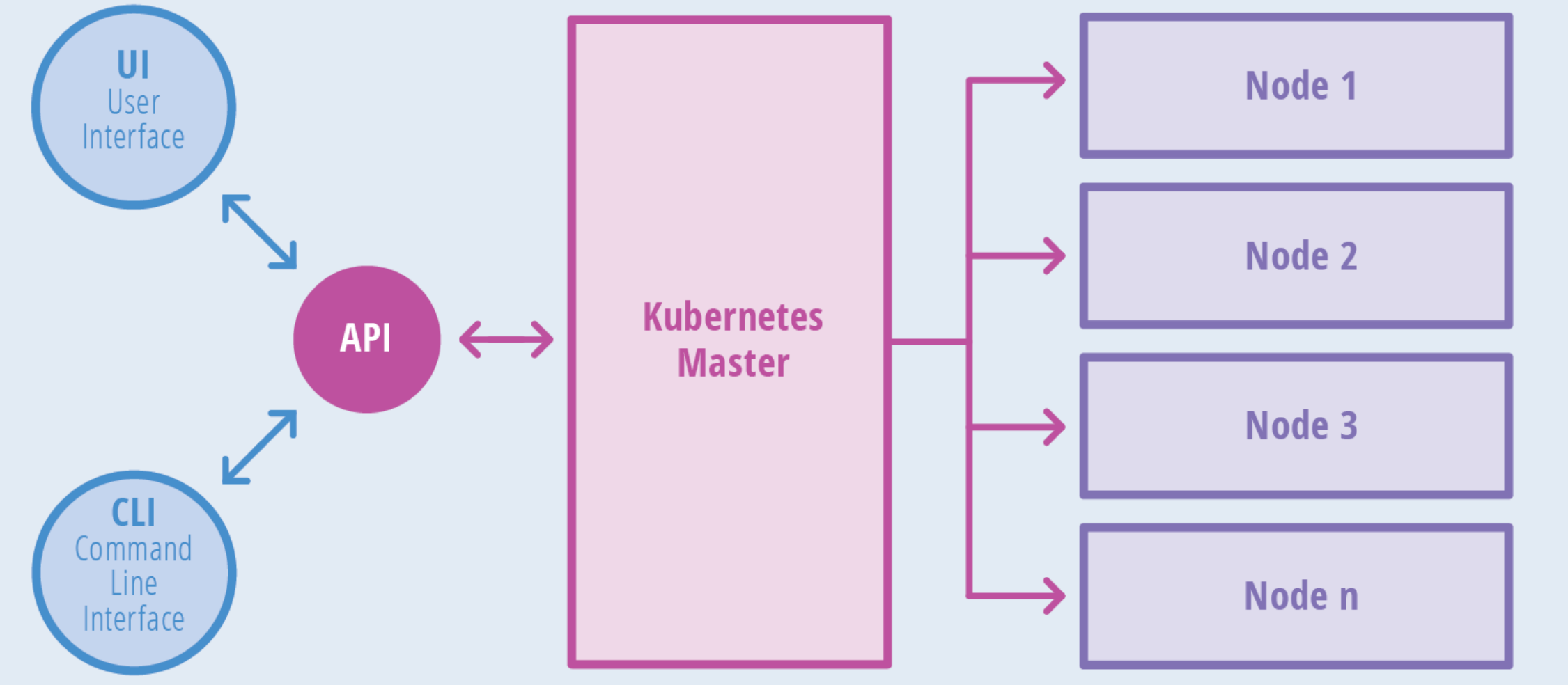
Kubernetes
What Docker looks like in production

Kubernetes
What Kubernetes looks like in production

Kubernetes
What is Kubernetes?

User Experience	argo, kubectl
Orchestration	VMware Tanzu, OPENSHIFT
Container Engine	docker, podman
Container Host	redhat LINUX

Kubernetes
What is Kubernetes?



Kubernetes
Live Demo

Live Demo

Kubernetes
Live Demo – Load-Test

```
docker run -rm technat/load-test -h https://alleaffengaffen.ch -c 5 -r 5m
```

Questions?

mweng@post.ch