WORKSHOP

STUD.IP & DOCKER

EIN VORTRAG VON

Florian Bieringer

Bytist

info@bytist.de

https://bytist.de

AGENDA

1. Was ist eigentlich Docker?
2. Docker Toolbox
3. Best Practises
4. Stud.IP Dockerimage
5. GitLab CI/CD
6. Beyond Docker
7. Fazit

DOCKER


MotivationalBuck.com
OVERWHELMED
Sure, I Can Handle The Load. No Problem.

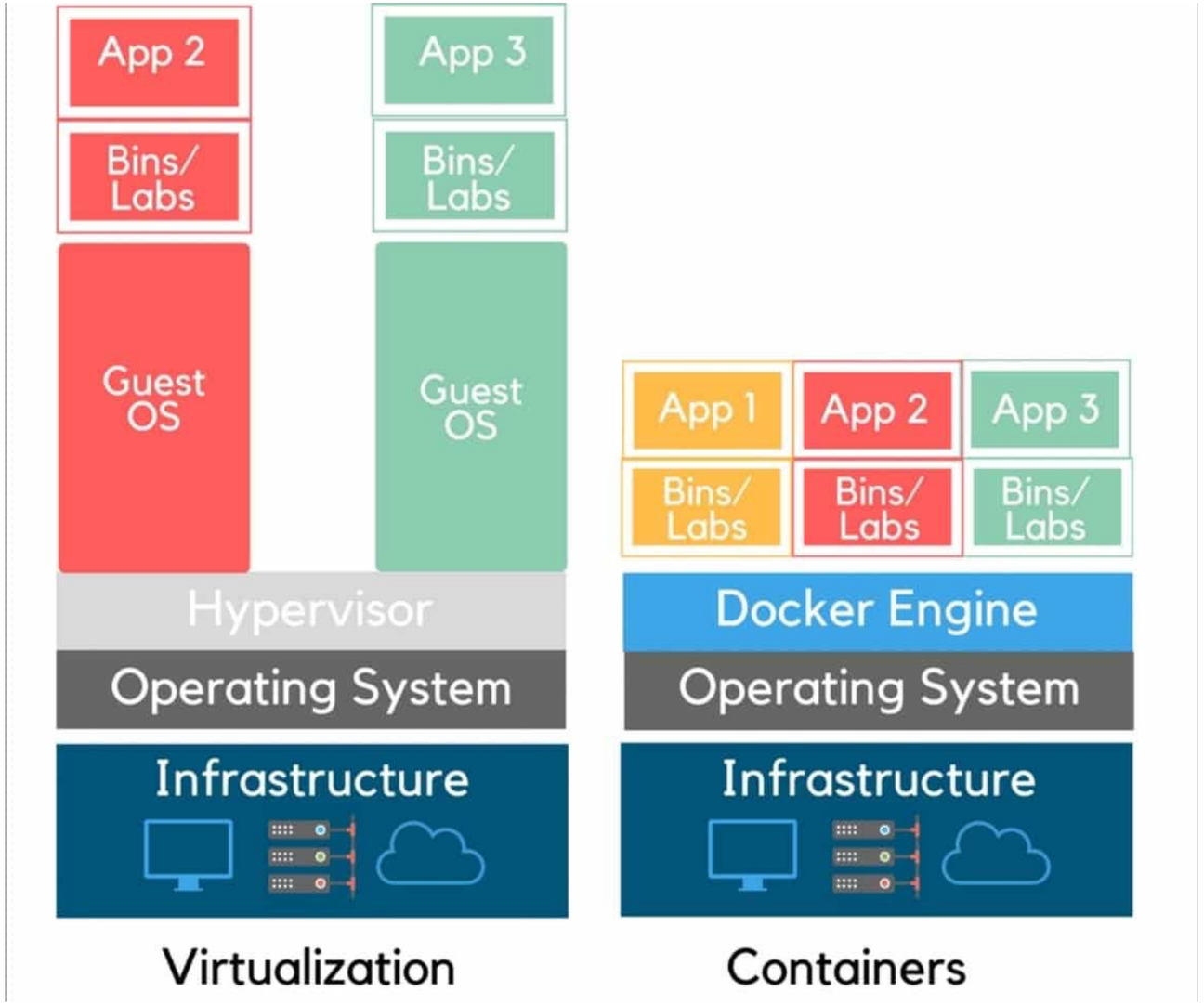
DOCKER

https://docs.docker.com

WAS IST EIGENTLICH

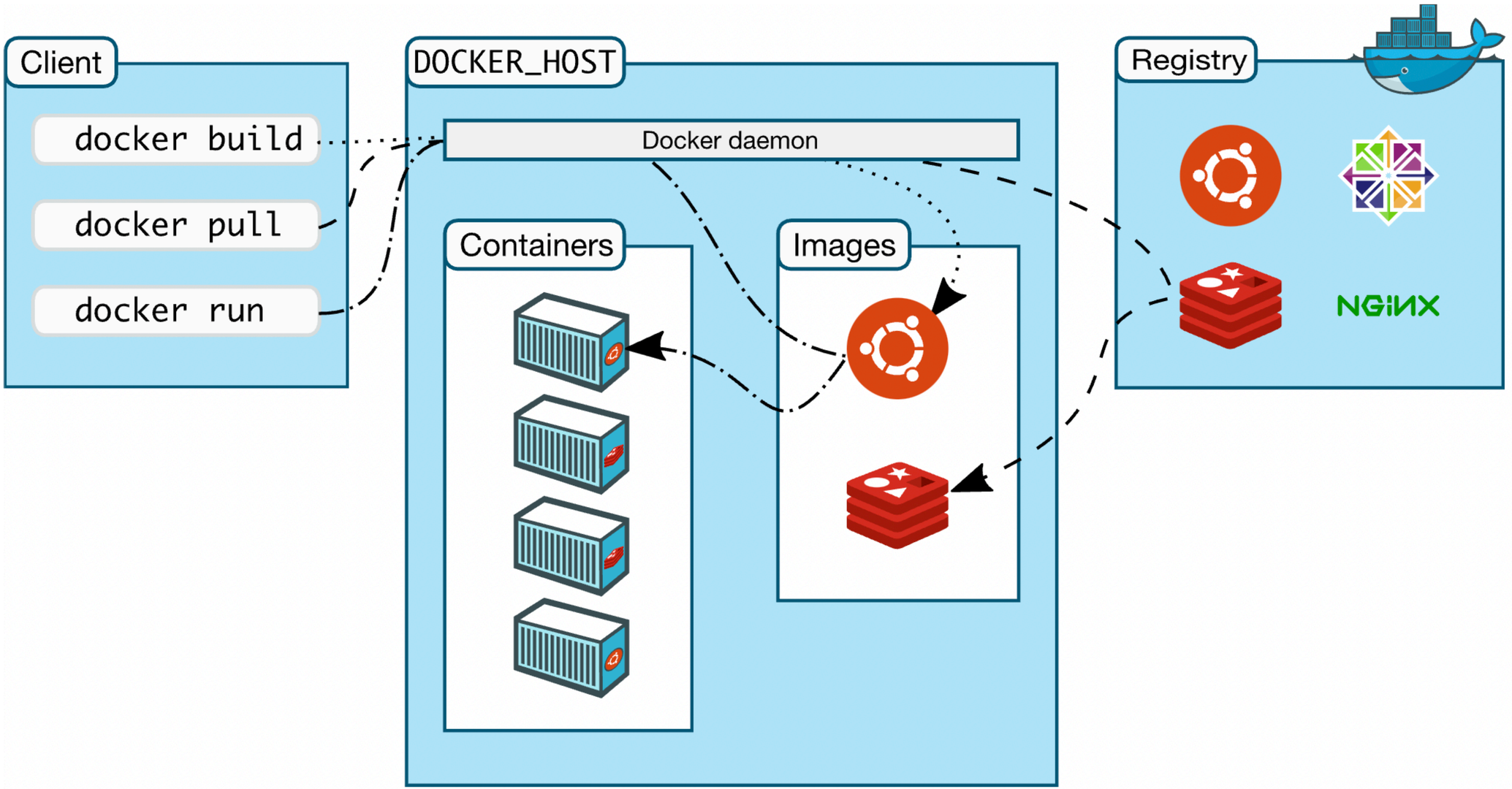
DOCKER?

DOCKER VS. VM


App 2
Bins/ Labs
Guest OS
App 3
Bins/ Labs
Guest OS
Hypervisor
Operating System
Infrastructure
Virtualization
App 1
App 2
App 3
Bins/ Labs
Bins/ Labs
Bins/ Labs
Docker Engine
Operating System
Infrastructure
Containers

VON UNIX ZU DOCKER

- Chroot
  - Virtuelles Stammverzeichnis für Programme
- LXC
  - Virtuelle Maschine (Host Kernel + SELinux + cgroups + …)
- Docker
  - Ursprünglich LXC Basis, heute libcontainer


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

DOCKER BESTANDTEILE

- Dockerfile
- Dockerimage
- Registry
- Dockercontainer

DOCKERFILE

```
# syntax=docker/dockerfile:1
FROM node:12-alpine
RUN apk add --no-cache python g++ make
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
```

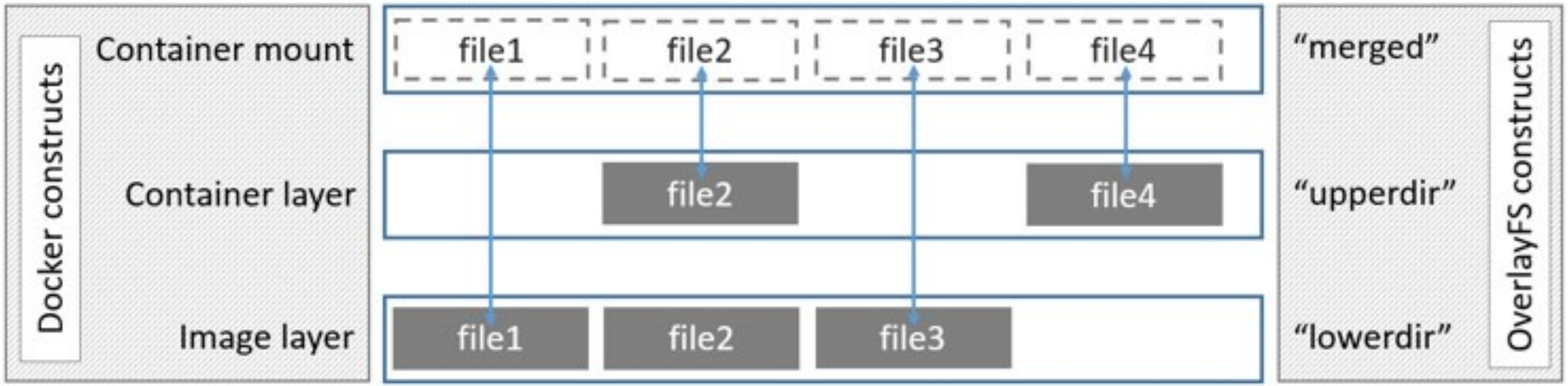
DOCKER IMAGES

- Gebaut durch Dockerfile (docker build .)
- Jede Anweisung erzeugt einen Layer
- Beinhalten nur Teile, die zur Ausführung benötigt werden
- Tagging ([registry/][namespace/]project:version)
- Entrypoint und CMD
- Interaction mit Registry über Docker Login/Push/Pull

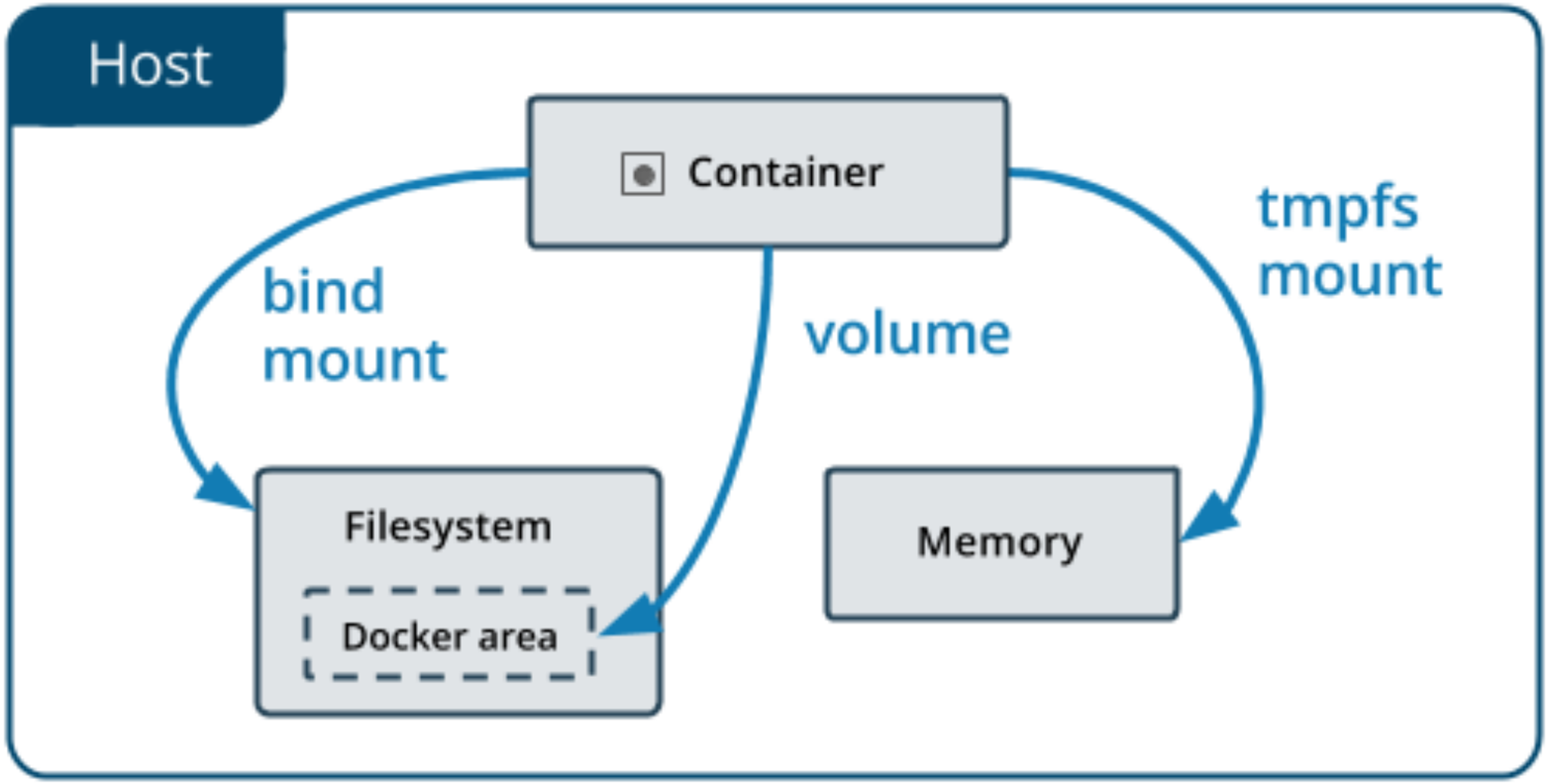
DOCKER CONTAINER

- Instanz
  - Gestartet durch „docker run [imagename]“
- Docker Image als Basis
  - Copy on write
- Volumes als persistente Speicherbereiche

OVERLAY


Docker constructs
Container mount
Container layer
Image layer
file1
file2
file3
file4
file2
file4
file1
file2
file3
"merged"
"upperdir"
"lowerdir"
OverlayFS constructs

DOCKER VOLUMES



DOCKER VOLUMES

- Bind Mount
  - Bindet Ordner des Hostsystems in den Container ein
- Docker Volume
  - Ähnlich wie Bind Mount, aber gemanaged durch Docker
- Tmpfs Mount
  - Mount im RAM + Swap

REGISTRY

- Verwaltung von Docker Images
- http://hub.docker.com als default registry
  - Docker run eines lokal nicht vorhanden Image lädt dies aus der Registry
  - Lokal vorhandene Images werden nicht automatisch geupdatet!
- Registry als Dockerimage https://hub.docker.com/_/registry

REGISTRY

- Verwaltung von Docker Images
- http://hub.docker.com als default registry
  - Docker run eines lokal nich es aus der Registry
  - Lokal vorhandene Images geupdatet!
- Registry als Dockerimage htt istry

DOCKER NETWORKING

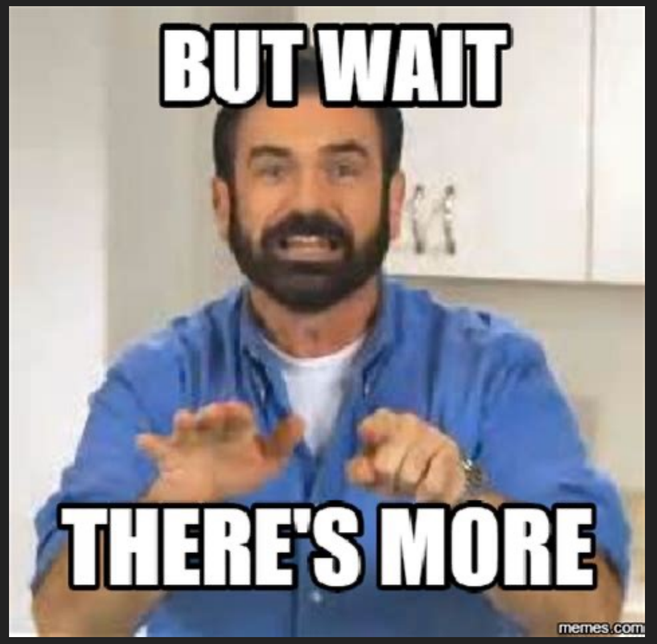
- Ports zur Freigabe von Diensten an den Host
- Per default ist ein Container von außen nicht erreichbar (NAT)
- Network: bridge, host, none, overlay, macvlan

DOCKER CONTAINER

- Docker Run
  - Startet Container

DOCKER CONTAINER

- Docker Run
  - Startet Container



DOCKER CONTAINER

- Docker Run
  - Startet Container

```
[flobieringer@MacBook-Air ~ % docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options:
      --add-host list                  Add a custom host-to-IP mapping (host:ip)
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16            Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
      --blkio-weight-device list       Block IO weight (relative device weight) (default [])
      --cap-add list                   Add Linux capabilities
      --cap-drop list                  Drop Linux capabilities
      --cgroup-parent string           Optional parent cgroup for the container
      --cgroupns string                Cgroup namespace to use (host|private)
                                       'host':    Run the container in the Docker host's cgroup namespace
                                       'private': Run the container in its own private cgroup namespace
                                       '':        Use the cgroup namespace as configured by the
                                                  default-cgroupns-mode option on the daemon (default)
      --cidfile string                 Write the container ID to the file
      --cpu-period int                 Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int                  Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int              Limit CPU real-time period in microseconds
      --cpu-rt-runtime int             Limit CPU real-time runtime in microseconds
  -c, --cpu-shares int                 CPU shares (relative weight)
      --cpus decimal                   Number of CPUs
      --cpuset-cpus string             CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string             MEMs in which to allow execution (0-3, 0,1)
  -d, --detach                         Run container in background and print container ID
      --detach-keys string             Override the key sequence for detaching a container
      --device list                    Add a host device to the container
      --device-cgroup-rule list        Add a rule to the cgroup allowed devices list
      --device-read-bps list           Limit read rate (bytes per second) from a device (default [])
      --device-read-iops list          Limit read rate (IO per second) from a device (default [])
      --device-write-bps list          Limit write rate (bytes per second) to a device (default [])
      --device-write-iops list         Limit write rate (IO per second) to a device (default [])
      --disable-content-trust          Skip image verification (default true)
      --dns list                       Set custom DNS servers
      --dns-option list                Set DNS options
      --dns-search list                Set custom DNS search domains
      --domainname string              Container NIS domain name
      --entrypoint string              Overwrite the default ENTRYPOINT of the image
  -e, --env list                       Set environment variables
      --env-file list                  Read in a file of environment variables
      --expose list                    Expose a port or a range of ports
      --gpus gpu-request               GPU devices to add to the container ('all' to pass all GPUs)
      --group-add list                 Add additional groups to join
      --health-cmd string              Command to run to check health
      --health-interval duration       Time between running the check (ms|s|m|h) (default 0s)
      --health-retries int             Consecutive failures needed to report unhealthy
      --health-start-period duration   Start period for the container to initialize before starting health-retries countdown (ms|s|m|h) (default 0s)
      --health-timeout duration        Maximum time to allow one check to run (ms|s|m|h) (default 0s)
      --help                           Print usage
  -h, --hostname string                Container host name
      --init                           Run an init inside the container that forwards signals and reaps processes
  -i, --interactive                    Keep STDIN open even if not attached
      --ip string                      IPv4 address (e.g., 172.30.100.104)
      --ip6 string                     IPv6 address (e.g., 2001:db8::33)
      --ipc string                     IPC mode to use
      --isolation string               Container isolation technology
      --kernel-memory bytes            Kernel memory limit
  -l, --label list                     Set meta data on a container
      --label-file list                Read in a line delimited file of labels
      --link list                      Add link to another container
      --link-local-ip list             Container IPv4/IPv6 link-local addresses
      --log-driver string              Logging driver for the container
      --log-opt list                   Log driver options
      --mac-address string             Container MAC address (e.g., 92:d0:c6:0a:29:33)
  -m, --memory bytes                   Memory limit
      --memory-reservation bytes       Memory soft limit
      --memory-swap bytes              Swap limit equal to memory plus swap: '-1' to enable unlimited swap
      --memory-swappiness int          Tune container memory swappiness (0 to 100) (default -1)
      --mount mount                    Attach a filesystem mount to the container
      --name string                    Assign a name to the container
      --network network                Connect a container to a network
      --network-alias list             Add network-scoped alias for the container
      --no-healthcheck                 Disable any container-specified HEALTHCHECK
      --oom-kill-disable               Disable OOM Killer
      --oom-score-adj int              Tune host's OOM preferences (-1000 to 1000)
      --pid string                     PID namespace to use
      --pids-limit int                 Tune container pids limit (set -1 for unlimited)
      --platform string                Set platform if server is multi-platform capable
      --privileged                     Give extended privileges to this container
  -p, --publish list                   Publish a container's port(s) to the host
  -P, --publish-all                    Publish all exposed ports to random ports
      --pull string                    Pull image before running ("always"|"missing"|"never") (default "missing")
      --read-only                      Mount the container's root filesystem as read only
      --restart string                 Restart policy to apply when a container exits (default "no")
      --rm                             Automatically remove the container when it exits
      --runtime string                 Runtime to use for this container
      --security-opt list              Security Options
      --shm-size bytes                 Size of /dev/shm
      --sig-proxy                      Proxy received signals to the process (default true)
      --stop-signal string             Signal to stop a container (default "SIGTERM")
      --stop-timeout int               Timeout (in seconds) to stop a container
      --storage-opt list               Storage driver options for the container
      --sysctl map                     Sysctl options (default map[])
      --tmpfs list                     Mount a tmpfs directory
  -t, --tty                            Allocate a pseudo-TTY
      --ulimit ulimit                  Ulimit options (default [])
  -u, --user string                    Username or UID (format: <name|uid>[:<group|gid>])
      --userns string                  User namespace to use
      --uts string                     UTS namespace to use
  -v, --volume list                    Bind mount a volume
      --volume-driver string           Optional volume driver for the container
      --volumes-from list              Mount volumes from the specified container(s)
  -w, --workdir string                 Working directory inside the container
flobieringer@MacBook-Air ~ %
```

DOCKER IST . . .

- Performant, da im Kernel des Hosts
- Versionierbar über Image über Dockerfile
- Einfach, wenn Image gut vorbereitet
- Sicher, da klar vom Host getrennt
- Überall einsetzbar (Läuft Stud.IP eigentlich auf Ubuntu xy?)

DOCKER

TOOLBOX

DOCKER COMPOSE

- Docker Run dargestellt als yml (docker-compose.yml)
- Docker-compose als Wrapper für docker Befehle

DOCKER COMPOSE

```yaml
version: "2"
services:
  db:
    image: mariadb:10.4
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_RANDOM_ROOT_PASSWORD: 1
      MYSQL_DATABASE: studip_db
      MYSQL_USER: studip_user
      MYSQL_PASSWORD: studip_password
  web:
    image: studip/studip:latest
    depends_on:
      - db

    # Use port to redirect port
    # ports:
    #   - "8032:80"

    restart: always
    environment:
      MYSQL_DATABASE: studip_db
      MYSQL_USER: studip_user
      MYSQL_PASSWORD: studip_password
      MYSQL_HOST: db

      # Use automigrate to migrate your instance on startup
      # AUTO_MIGRATE: 1

      # Use proxy url OR autoproxy if run behind a proxy
      # PROXY_URL: https://studip.example.com/
      # AUTO_PROXY: 1

      # Demo data for your studip instance
      DEMO_DATA: 1

volumes:
  db_data: {}
```

DOCKER COMPOSE

- Environment als wichtiger Bestandteil des Deployments

```yaml
version: "2"
services:
  db:
    image: mariadb:10.4
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_RANDOM_ROOT_PASSWORD: 1
      MYSQL_DATABASE: studip_db
      MYSQL_USER: studip_user
      MYSQL_PASSWORD: studip_password
```

DOCKER NETWORKING

- Ports zur Freigabe von Diensten an den Host
- Per default ist ein Container von außen nicht erreichbar
- Network: host
- Docker-compose erzeugt default Netzwerk (+DNS) für alle Container

```
  web:
    image: studip/studip:latest
    depends_on:
      - db

    # Use port to redirect port
    # ports:
    #   - "8032:80"
```

DOCKER CLEANUP

- Docker system prune
  - Löscht alle nicht verwendeten Teile von Docker

DOCKER SHELL

- Docker-compose exec [servicename] /bin/bash
  - Startet eine Bash im Container (wenn vorhanden, alternativ: /bin/sh)
  - Container muss gestartet sein, sonst „run“ verwenden
  - VSCode erlaubt auch Dateibrowser in Container
  - Shell als Ultima Ratio

BEST

PRACTICES

ONE JOB PER CONTAINERS

- Container haben keinen „init“ Prozess
  - Tini als Alternative (eher nein)
  - Kommandos im Hintergrund (z.B. „cron -f &“) als Negativbeispiel
- Pro Programm einen Container
- Log nach stdout / stderr zur Weiterverwendung in Docker Log

CREATE EPHEMERAL CONTAINERS

- Container soll im Standardzustand laufen
- Container kann jederzeit ersetzt werden
  - Auch Updates sollen funktionieren

IMAGES SO KLEIN WIE MÖGLICH

- .dockerignore um unnötige Teile wegzulassen
- Keine unnötigen Pakete installieren
- Multi Stage Builds
- Alpine Image benutzen

WEITERES

- RUN erzeugt einen Layer!
- Mehrzeilige Anweisungen verwenden
- Anweisungen alphabetisch sortieren

```
RUN apt-get update && apt-get install -y \
    package-bar \
    package-baz \
    package-foo  \
    && rm -rf /var/lib/apt/lists/*
```

DEPLOYMENT

- Bind Mounts vermeiden
- Modifikationen am Dockerimage als eignes Dockerfile
- Nur notwendige Ports nach außen geben
- Verwendung des Environments

STUD.IP

DOCKERIMAGE

PROPOSAL

- Je ein Dockerimage pro Minor Version
  - Latest = Letzte aktuelle Version
  - Nightly = Main
- Je Basis: php-apache + php-fpm-alpine
- Autobuild + Push über GitLab CI
- Dockerimages für Testing separat

PROPOSAL

- Docker Informationen direkt im Stud.IP Git
- https://hub.docker.com/r/studip/studip
  - Angelegt als „Organization“
- Vielleicht zukünftig: Official Docker Image

UBUNTU OFFICIAL IMAGE

ubuntu

Official Image

Ubuntu is a Debian-based Linux operating system based on free software.

1B+

Container | Linux | IBM Z | ARM | riscv64 | PowerPC 64 LE | ARM 64 | x86-64 | 386 | Base Images

Operating Systems | Official Image

Copy and paste to pull this image

```
docker pull ubuntu
```

View Available Tags

STUD.IP APACHE IMAGE

```
# Setup php, apache and stud.ip
FROM php:7.4-apache as base

# Install system requirements
RUN apt update && apt install -y --no-install-recommends \
        default-mysql-client \
        default-libmysqlclient-dev \
        libcurl4-openssl-dev zlib1g-dev \
        libpng-dev \
        libjpeg-dev \
        libonig-dev \
        libzip-dev \
        libicu-dev \
    && rm -rf /var/lib/apt/lists/*

# Install php extensions
RUN docker-php-ext-configure gd --with-jpeg
RUN docker-php-ext-install pdo gettext curl gd mbstring zip pdo pdo_mysql mysqli intl json
```

STUD.IP APACHE IMAGE

```
19
20  FROM node:12 as nodejs
21
22  # Install node modules
23  COPY . /studip
24  WORKDIR /studip
25  RUN npm install && npm run webpack-prod
26
27  FROM base as build
28
29  # Install composer
30  COPY --from=composer /usr/bin/composer /usr/bin/composer
31
32  # Copy studip
33  COPY --from=nodejs /studip /studip
34
35  # Execute make to install composer dependencies and build assets
36  WORKDIR /studip
37  RUN composer install --no-dev --optimize-autoloader
38
```

STUD.IP APACHE IMAGE

```
39 FROM base
40
41 # Reconfigure apache
42 ENV APACHE_DOCUMENT_ROOT /var/www/studip/public
43 RUN sed -ri -e 's!/var/www/html!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/sites-available/*.conf
44 RUN sed -ri -e 's!/var/www/!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/apache2.conf /etc/apache2/conf-available/*.conf
45
46 COPY --from=build /studip /var/www/studip
47
48 WORKDIR /var/www/studip
49
50 # Add config template
51 COPY ./docker/apache/config_local.php /config/config_local.inc.php
52
53 # Add custom entrypoint
54 COPY ./docker/apache/docker-entrypoint.sh /usr/local/bin/
55 RUN chmod u+x /usr/local/bin/docker-entrypoint.sh
56
57 # Set start parameters
58 ENTRYPOINT ["/usr/local/bin/docker-entrypoint.sh"]
59 CMD ["apache2-foreground"]
```

STUD.IP APACHE IMAGE

```php
<?php
/*basic settings for Stud.IP
-----------------------------------------------------------------
you find here the basic system settings. You shouldn't have to touch much of them...
please note the CONFIG.INC.PHP for the indivual settings of your installation!*/

namespace Studip {
    //const ENV = 'development';
    define ('ENV', getenv('ENV') ?? 'development');
}

namespace {
    /*settings for database access
    ----------------------------------------------------------------
    please fill in your database connection settings.
    */

    // default Stud.IP database (DB_Seminar)
    $DB_STUDIP_HOST = getenv('MYSQL_HOST');
    $DB_STUDIP_USER = getenv('MYSQL_USER');
    $DB_STUDIP_PASSWORD = getenv('MYSQL_PASSWORD');
    $DB_STUDIP_DATABASE = getenv('MYSQL_DATABASE');
```

STUD.IP APACHE IMAGE

```
24      /*URL
25      ---------------------------------------------------------------
26      customize if automatic detection fails, e.g. when installation is hidden
27      behind a proxy
28      */
29      //$CANONICAL_RELATIVE_PATH_STUDIP = '/';
30      //$ABSOLUTE_URI_STUDIP = 'https://www.studip.de/';
31      //$ASSETS_URL = 'https://www.studip.de/assets/';
32
33      // Set proxy url
34      if ($PROXY_URL = getenv('PROXY_URL')) {
35          $ABSOLUTE_URI_STUDIP = $PROXY_URL;
36          $ASSETS_URL = $PROXY_URL.'/assets/';
37          unset($PROXY_URL);
38      }
39
40      // Use autoproxy
41      if (getenv('AUTO_PROXY')) {
42          $ABSOLUTE_URI_STUDIP = $_SERVER['HTTP_X_FORWARDED_PROTO'].'://'.$_SERVER['HTTP_X_FORWARDED_HOST'].'/';
43          $ASSETS_URL = $ABSOLUTE_URI_STUDIP.'/assets/';
44      }
45  }
```

STUD.IP APACHE IMAGE

docker-entrypoint.sh 2.06 KB

```
#!/bin/bash
set -e

STUDIP='/var/www/studip'
CONFIGFILE="$STUDIP/config/config_local.inc.php"
DOCKERCONFIGFILE="/config/config_local.inc.php"
CONF="$STUDIP/config/config.inc.php"

# Check if we have a config
if [ ! -f $CONFIGFILE ]; then
    echo "Setting up new config"

    cp "$DOCKERCONFIGFILE" "$CONFIGFILE"
    cp "$CONF.dist" "$CONF"
fi
```

STUD.IP APACHE IMAGE

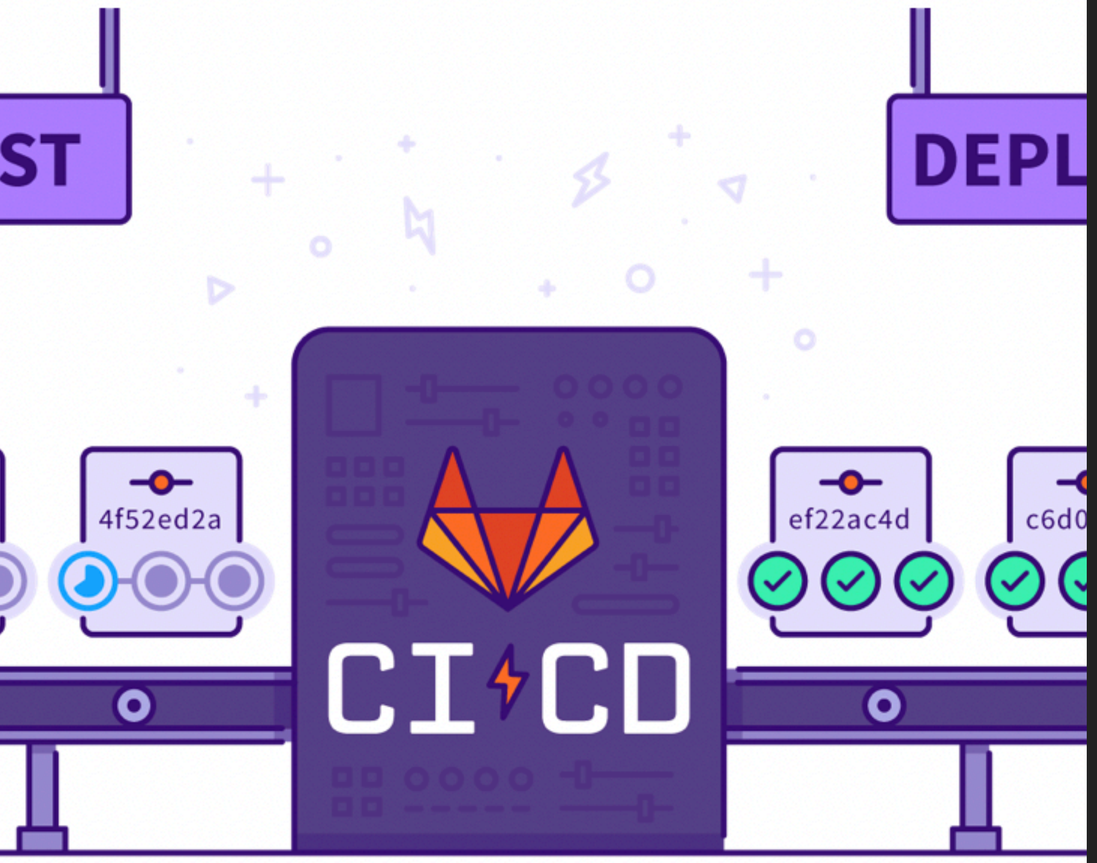
```
16
17  # wait until MySQL is really available
18  maxcounter=45
19
20  counter=1
21  while ! mysql -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD -e "show databases;" > /dev/null 2>&1; do
22      sleep 1
23      counter=`expr $counter + 1`
24      if [ $counter -gt $maxcounter ]; then
25          >&2 echo "We have been waiting for MySQL too long already; failing."
26          exit 1
27      fi;
28  done
```

STUD.IP APACHE IMAGE

```
30  # Check if the connected database has tables, otherwise install the database
31  if [[ $(mysql -f -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD $MYSQL_DATABASE -e "show tables;" --batch | wc -l) -eq 0 ]]; then
32
33      # Setup mysql database
34      echo "INSTALL DB"
35      mysql -f -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD $MYSQL_DATABASE < ${STUDIP}/db/studip.sql
36      echo "INSTALL DEFAULT DATA"
37      mysql -f -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD $MYSQL_DATABASE < ${STUDIP}/db/studip_default_data.sql
38      mysql -f -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD $MYSQL_DATABASE < ${STUDIP}/db/studip_resources_default_data.sql
39
40      echo "INSTALL ROOTUSER"
41      mysql -f -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD $MYSQL_DATABASE < ${STUDIP}/db/studip_root_user.sql
42
43      # Check if demodata is required
44      if [ ! -z $DEMO_DATA ]; then
45          echo "INSTALL DEMODATA"
46          mysql -f -u $MYSQL_USER -h $MYSQL_HOST -p$MYSQL_PASSWORD $MYSQL_DATABASE < ${STUDIP}/db/studip_demo_data.sql
47      fi
48
49      echo "INSTALLATION FINISHED"
50  else
51      echo "Found some SQL table. Skipping installation"
52  fi
53
```

STUD.IP APACHE IMAGE

```
53
54  if [ ! -z $AUTO_MIGRATE ]; then
55      echo "Migrate Instance"
56
57      # If migrate fails start instance anyway
58      php "$STUDIP/cli/migrate.php" || true
59      echo "Migration finished"
60  fi
61
62  # first arg is `-f` or `--some-option`
63  if [ "${1#-}" != "$1" ]; then
64          set -- apache2-foreground "$@"
65  fi
66
67  exec "$@"
```



GITLAB

CI/CD

WARUM GITLAB CI?

- Am besten integriert in Gitlab
- Vorreiterrolle (viele Features unterstützt)
- GitLab Runner als eigenständiges Softwarepaket

PIPELINES

- Pipelines führen Aufgaben durch (Build, Test, Deploy)
- Automatisch via .gitlab-ci.yml
- Sehr gut dokumentiert seitens Gitlab
- Automatisch gestartet über Commit (Push)

.GITLAB-CI.YML

```yaml
stages:
  - test
  - build

# Tests for php7.2
test_php7.2:
  image: studip/studip:tests-php7.2
  stage: test
  allow_failure: true
  script:

    # Copy docker config
    - cp ./docker/tests/config_local.php ./config/config_local.inc.php
    - cp ./config/config.inc.php.dist ./config/config.inc.php

    # Execute tests
    - make test

```

GITLAB RUNNER

- Führt Pipelines aus
- Kommuniziert mit GitLab via API (Betrieb hinter NAT möglich)
- Unterstützt verschiedene Executors

EXECUTORS

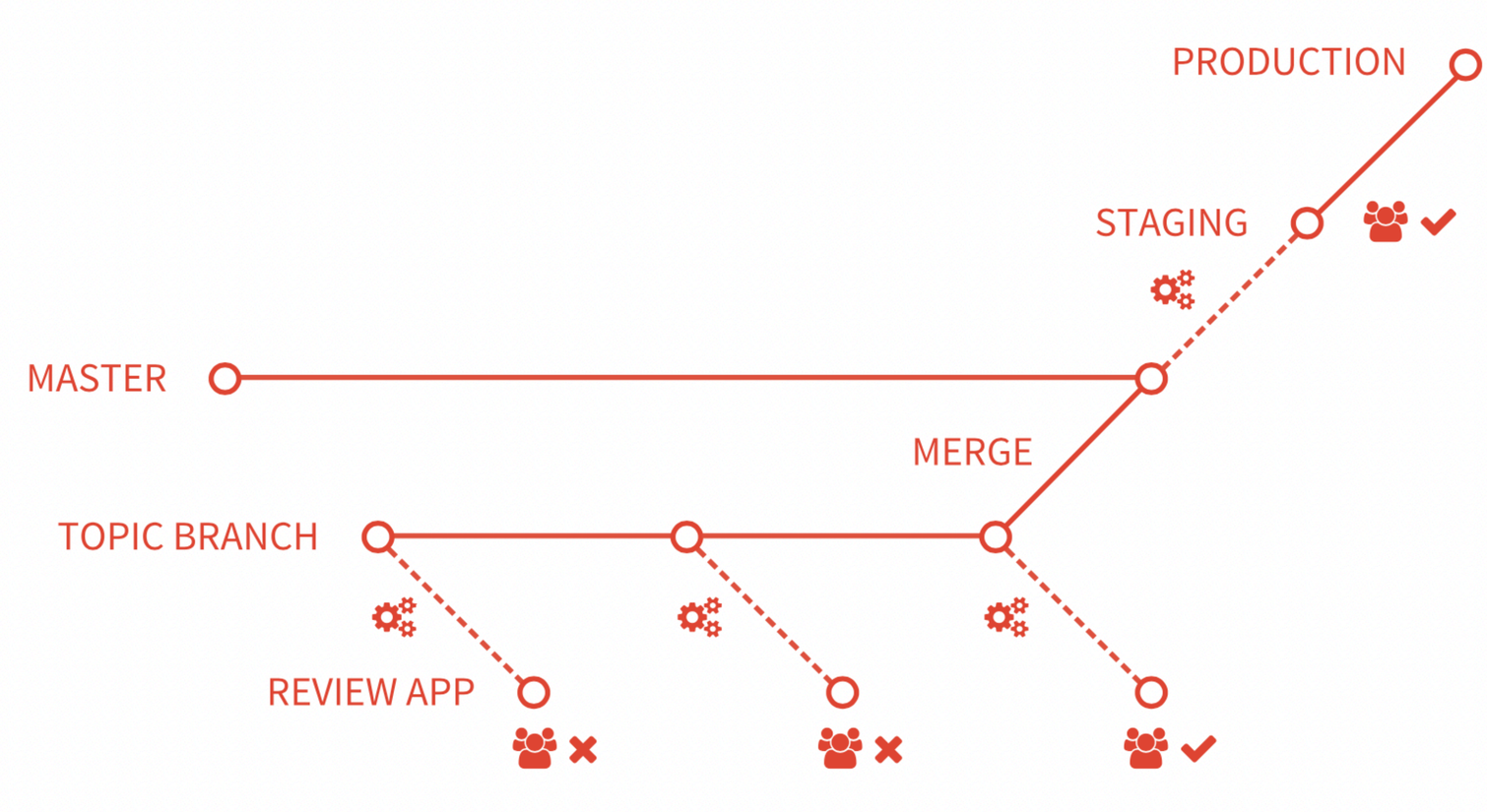
Executor	SSH	Shell	VirtualBox	Parallels	Docker	Kubernetes	Custom
Clean build environment for every build	✗	✗	✓	✓	✓	✓	conditional (4)
Reuse previous clone if it exists	✓	✓	✗	✗	✓	✗	conditional (4)
Runner file system access protected (5)	✓	✗	✓	✓	✓	✓	conditional
Migrate runner machine	✗	✗	partial	partial	✓	✓	✓
Zero-configuration support for concurrent builds	✗	✗ (1)	✓	✓	✓	✓	conditional (4)
Complicated build environments	✗	✗ (2)	✓ (3)	✓ (3)	✓	✓	✓
Debugging build problems	easy	easy	hard	hard	medium	medium	medium

1. It's possible, but in most cases it is problematic if the build uses services installed on the build machine
2. It requires to install all dependencies by hand
3. For example using Vagrant
4. Dependent on what kind of environment you are provisioning. It can be completely isolated or shared between each build.
5. When a runner's file system access is not protected, jobs can access the entire system including the runner's token, and the cache and code of other jobs. Executors marked ✓ don't allow the runner to access the file system by default. However, security flaws or certain configurations could allow jobs to break out of their container and access the file system hosting runner.

GITLAB RUNNER MIT DOCKER EXECUTOR

- Geschützte Umgebung durch Docker Sandbox
- Saubere Umgebung
- Umgebung nachvollziehbar durch Dockerfile
- Mehrere Stages parallelisierbar (z.B. PHP7.4 + PHP8 Unittests)
- Skalierung der Runner Instanzen ohne Probleme auf mehrere Server und Standorte

REVIEW APPS



BEYOND

DOCKER

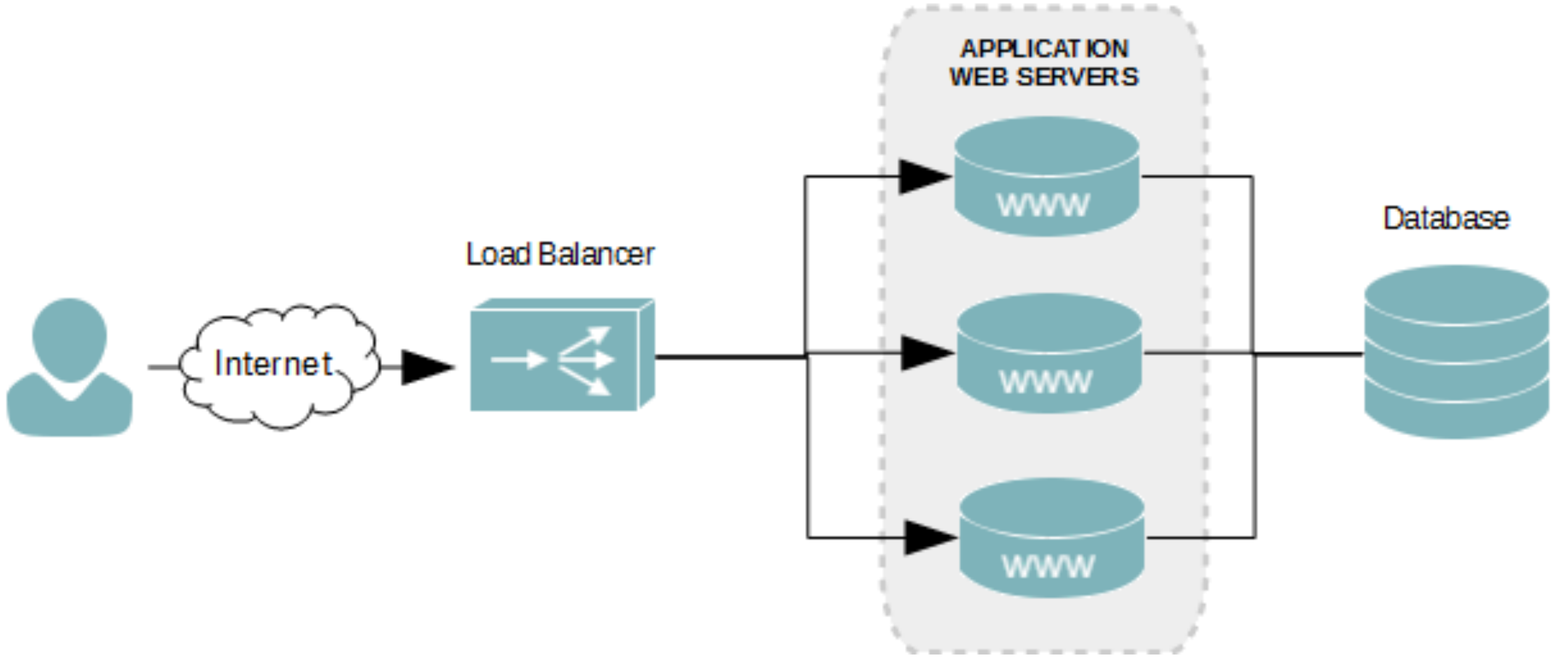
ORCHESTRIERUNG

- Docker bietet einen standardisierten Service, der leicht weitere Container hinzufügen kann
  - Skalierung
  - Ausfallsicherheit
- Kubernetes, Dockerswarm, Mesos, …

ORCHESTRIERUN

- Docker bietet Container hinzufügen ka
  - Skalierung
  - Ausfallsiche
- Kubernetes, D

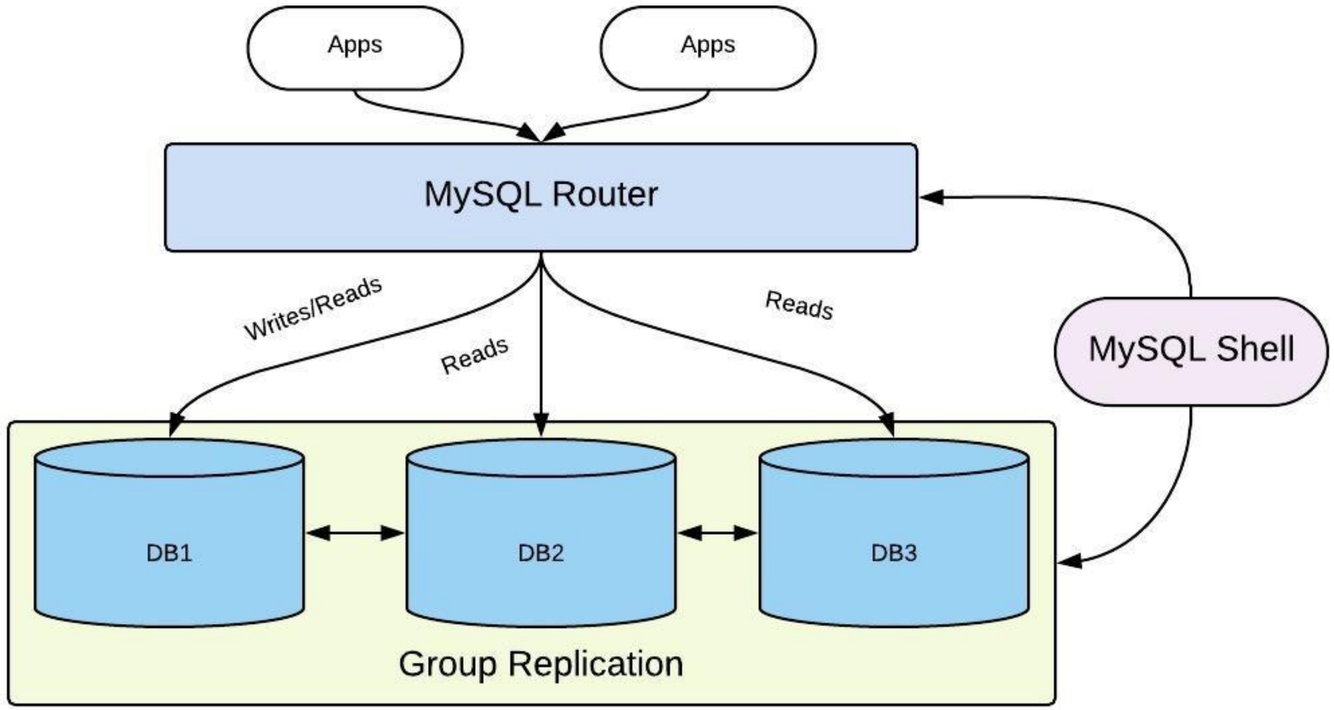
SKALIERUNG WEBAPPS



PROBLEME

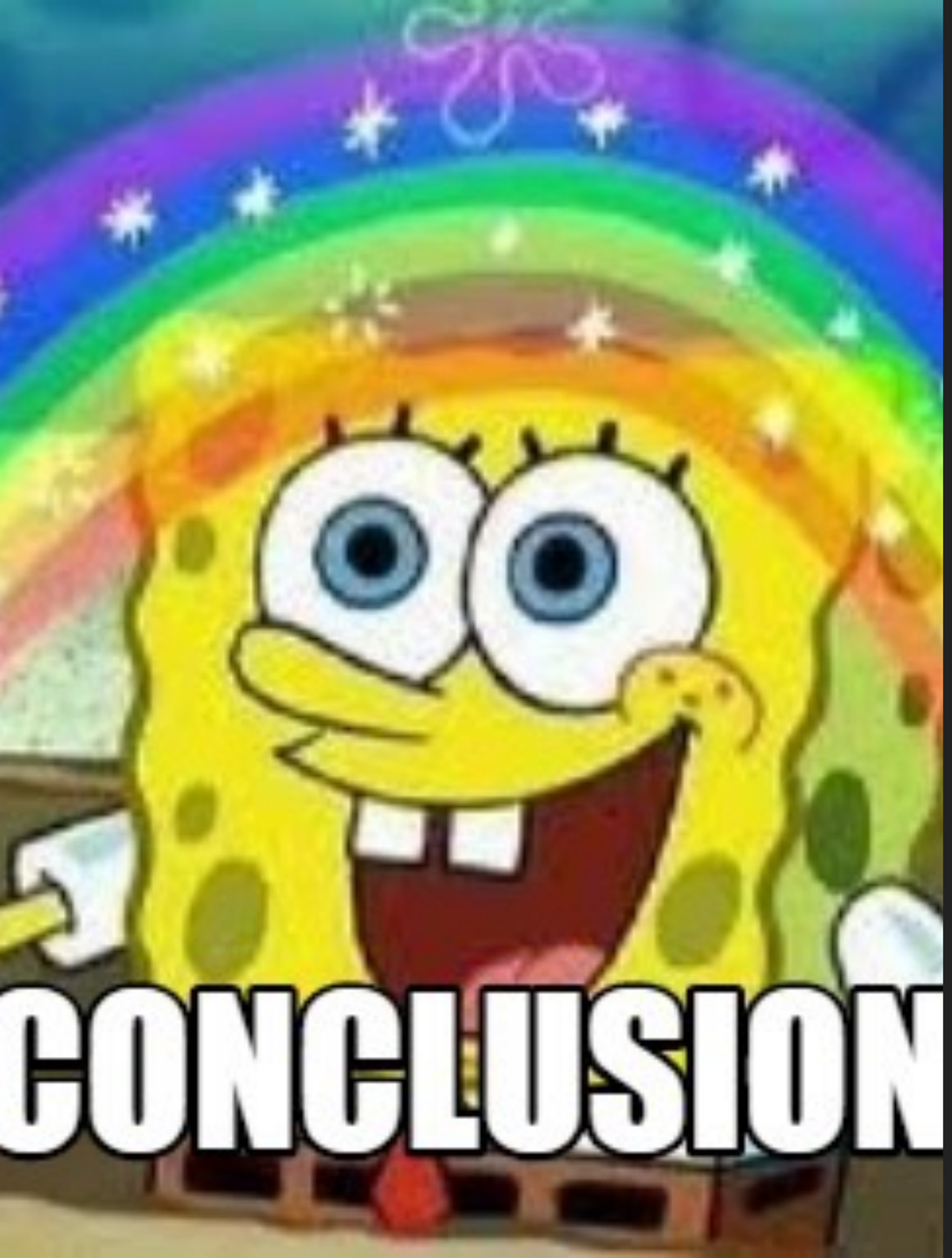
- Loadbalancer absichern über IP Switching
- Datenbankcluster immer noch in den Kinderschuhen
  - Aber große Fortschritte in den letzen Jahren dank Docker
- Verteiltes Dateisystem für Daten (z.B. GlusterFS, Ceph, …)
  - Vorteil: Viele Systeme bringen bereits einen Dockerdriver mit

INNODB CLUSTER



LOHNT SICH EINE MIGRATION IN EINEN CLUSTER FÜR MICH?

- Was stellt mein RZ zur Verfügung?
  - Managed Docker Swarm / Kubernetes
  - Loadbalancer
  - Datenbankservice
  - Distributed Storage



FAZIT

WAS LEISTET DOCKER?

- Vorbereitetes Stud.IP spart Zeit für Neulinge und kleine Installationen
- Keine Probleme mit Host PHP oder mySQL Versionen
- Docker Container Overhead verschwindend gering
- Image als Basis für verteilte, hochverfügbare Systeme
- Versionierung von Anpassungen des Originalimages
- Test von Commits über mehrere PHP Setups
- Automatische Review Umgebungen für Features