

Dockerbank II

**Vertiefungsworkshop zum Container-basierten
Deployment von biomedizinischen IT-Lösungen**

Block 2: Komplexbeispiel

Benjamin Baum¹, Christian Bauer¹

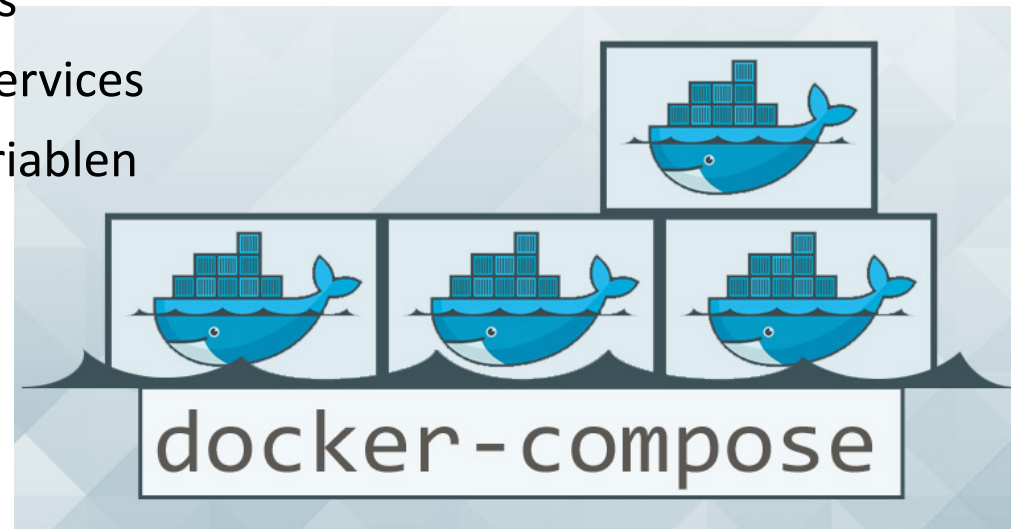
¹ Universitätsmedizin Göttingen

Definition

- ▶ Compose ist ein Tool zur Definition und zum Ausführen von Multi-Container-Docker-Applications.

Übersicht Eigenschaften

- ▶ Registrierung eigener Services
- ▶ Abhängigkeiten zu anderen Services
- ▶ Übergabe von Umgebungsvariablen
- ▶ Verlinkung von Containern
- ▶ Öffnen von Ports



<https://learning-continuous-deployment.github.io/dockercompose/multi-app/2015/05/30/docker-compose/>

Docker Compose

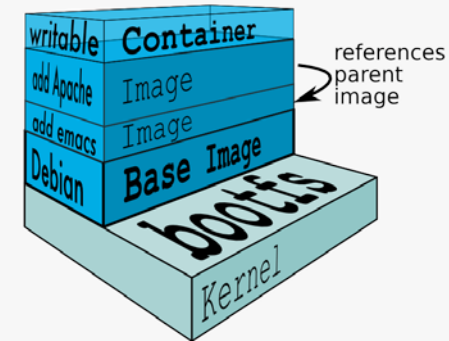


```
docker create -v /opt/pg/ --name transmart-db-data tmfev/transmart-db-data
```

```
docker run --name transmart-db --volumes-from transmart-db-data -e "POSTGRES_PASSWORD=POSTGRESSPASSWORDCHANGEME" -e "PGDATA=/opt/pg/data" -p 5432:5432 -d tmfev/transmart-db
```



```
version: '2'
services:
  transmartdbdata:
    image: tmfev/transmart-db-data
    volumes:
      - /opt/pg
  transmartdb:
    image: tmfev/transmart-db
    environment:
      - PGDATA:/opt/pg/data
      - POSTGRES_PASSWORD:docker
    ports:
      - "5432:5432"
    depends on:
```



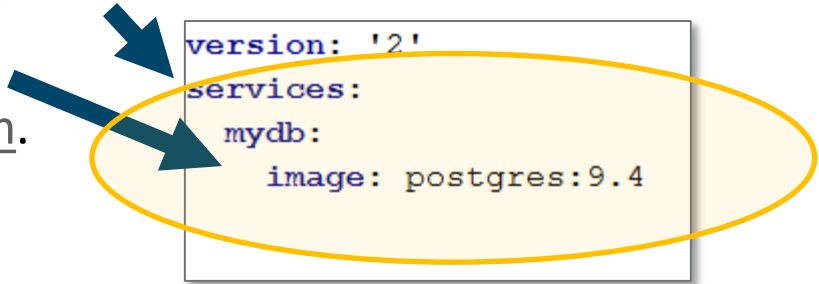
<https://blog.docker.com/2015/10/docker-basics-webinar-qa/>

Docker Compose

Definitionsdatei docker-compose.yml

- ▶ Definiert in einer YAML-Datei *docker-compose.yml*
- ▶ Definition eines service mit Namen *mydb*.
- ▶ Auswahl und Nutzung eines Docker-Containers von <https://hub.docker.com>.

```
version: '2'
services:
  mydb:
    image: postgres:9.4
```





OFFICIAL REPOSITORY

postgres ☆

Last pushed: 13 days ago

Repo Info

Tags

Short Description

The PostgreSQL object-relational database system provides reliability and data integrity.

Full Description

Supported tags and respective Dockerfile links

- [9.6.1](#), [9.6](#), [9](#), [latest](#) ([9.6/Dockerfile](#))
- [9.6.1-alpine](#), [9.6-alpine](#), [9-alpine](#), [alpine](#) ([9.6/alpine/Dockerfile](#))
- [9.5.5](#), [9.5](#) ([9.5/Dockerfile](#))
- [9.5.5-alpine](#), [9.5-alpine](#) ([9.5/alpine/Dockerfile](#))
- [9.4.10](#), [9.4](#) ([9.4/Dockerfile](#))
- [9.4.10-alpine](#), [9.4-alpine](#) ([9.4/alpine/Dockerfile](#))
- [9.3.15](#), [9.3](#) ([9.3/Dockerfile](#))
- [9.3.15-alpine](#), [9.3-alpine](#) ([9.3/alpine/Dockerfile](#))
- [9.2.19](#), [9.2](#) ([9.2/Dockerfile](#))
- [9.2.19-alpine](#), [9.2-alpine](#) ([9.2/alpine/Dockerfile](#))

Docker Pull Command



```
docker pull postgres
```

Docker Compose

Definitionsdatei docker-compose.yml

- ▶ Definiert in einer YAML-Datei `docker-compose.yml`
- ▶ Definition eines services mit Namen `mydb`
- ▶ Auswahl und Nutzung eines Docker-Containers von <https://hub.docker.com>.
- ▶ Oder Nutzung von eigenen, lokalen Containern ...
- ▶ ... mit eigener Konfiguration und Argumenten.

```
version: '2'
services:
  mydb:
    image: postgres:9.4
```

```
version: '2'
services:
  mydb:
    build: ..\MyDockerContainer
```

```
version: '2'
services:
  mydb:
    build:
      context: ..\MyDockerContainer
      dockerfile: MyDockerConfig
      args:
        password: sj39!2$sKo3
```

Docker Compose

Definitionsdatei docker-compose.yml

- ▶ Gerätefreigabe

- ▶ Verlinkung von Container
 - ▶ expose: Bereitstellen von Ports
 - ▶ links: Verknüpfung zweier Container

- ▶ Portfreigabe nach Außen

- ▶ Gemeinsame Dateibereiche
 - ▶ volumes
 - ▶ volumes_from

```
devices:
  - "/dev/ttyUSB0:/dev/ttyUSB0"
```

```
version: '2'
services:
  mydb:
    image: postgres:9.4
    expose: 5432
  myapp:
    image: myapp:latest
    links: mydb:mydb
```

```
ports:
  - "5432:5432"
```

```
services:
  mydb:
    image: postgres:9.4
    volumes:
      - /opt/test
```

```
volumes_from:
  - mydb
```

Docker Compose

Definitionsdatei docker-compose.yml

- ▶ Variablen
 - ▶ environment
 - ▶ env_file

```
environment:  
  - POSTGRES_PASSWORD:docker  
  - DROP:false  
  - UPLOAD:true
```

```
env_file: .env
```

- ▶ Reihenfolge

```
depends_on:  
  - transmartdb
```

- ▶ entrypoint

```
entrypoint: bash /opt/wait-for-pg.sh transmart-db
```


Docker Compose

docker-compose Befehle

▶ docker-compose build

```
Usage: build [options] [SERVICE...]

Options:
--force-rm  Always remove intermediate containers.
--no-cache  Do not use cache when building the image.
--pull      Always attempt to pull a newer version of the image.
```

▶ docker-compose create

```
Creates containers for a service.

Usage: create [options] [SERVICE...]

Options:
--force-recreate  Recreate containers even if their configuration and
                  image haven't changed. Incompatible with --no-recreate.
--no-recreate     If containers already exist, don't recreate them.
                  Incompatible with --force-recreate.
--no-build        Don't build an image, even if it's missing.
--build           Build images before creating containers.
```

Docker Compose

docker-compose Befehle

- ▶ docker-compose start / stop / restart / kill

```
Usage: start [SERVICE...]
```

```
Usage: stop [options] [SERVICE...]
```

Options:

```
-t, --timeout TIMEOUT    Specify a shutdown timeout in seconds (default: 10).
```

```
Usage: restart [options] [SERVICE...]
```

Options:

```
-t, --timeout TIMEOUT    Specify a shutdown timeout in seconds. (default: 10)
```

```
Usage: kill [options] [SERVICE...]
```

Options:

```
-s SIGNAL                SIGNAL to send to the container. Default signal is SIGKILL.
```

Docker Compose

docker-compose Befehle

- ▶ docker-compose start / stop / restart / kill / pause / unpause / run

```
Usage: pause [SERVICE...]
```

```
Usage: unpause [SERVICE...]
```

```
Usage: run [options] [-e KEY=VAL...] SERVICE [COMMAND] [ARGS...]
```

Options:

<code>-d</code>	Detached mode: Run container in the background, print new container name.
<code>--name NAME</code>	Assign a name to the container
<code>--entrypoint CMD</code>	Override the entrypoint of the image.
<code>-e KEY=VAL</code>	Set an environment variable (can be used multiple times)
<code>-u, --user=""</code>	Run as specified username or uid
<code>--no-deps</code>	Don't start linked services.
<code>--rm</code>	Remove container after run. Ignored in detached mode.
<code>-p, --publish=[]</code>	Publish a container's port(s) to the host
<code>--service-ports</code>	Run command with the service's ports enabled and mapped to the host.
<code>-T</code>	Disable pseudo-tty allocation. By default <code>docker-compose run</code> allocates a TTY.
<code>-w, --workdir=""</code>	Working directory inside the container

Docker Compose

docker-compose Befehle

- ▶ docker-compose start / stop / restart / kill / pause / unpause / run
- ▶ docker-compose up / down

```
Usage: up [options] [SERVICE...]

Options:
  -d                Detached mode: Run containers in the background,
                    print new container names.
                    Incompatible with --abort-on-container-exit.
  --no-color        Produce monochrome output.
  --no-deps         Don't start linked services.
  --force-recreate  Recreate containers even if their configuration
                    and image haven't changed.
                    Incompatible with --no-recreate.
  --no-recreate     If containers already exist, don't recreate them.
                    Incompatible with --force-recreate.
  --no-build        Don't build an image, even if it's missing.
  --build           Build images before starting containers.
  --abort-on-container-exit Stops all containers if any container was stopped.
                    Incompatible with -d.
  -t, --timeout TIMEOUT Use this timeout in seconds for container shutdown
                        when attached or when containers are already
                        running. (default: 10)
  --remove-orphans  Remove containers for services not defined in
                    the Compose file
```

Docker Compose

docker-compose Befehle

- ▶ docker-compose start / stop / restart / kill / pause / unpause / run
- ▶ docker-compose up / down

```
Usage: down [options]

Options:
  --rmi type           Remove images. Type must be one of:
                       'all': Remove all images used by any service.
                       'local': Remove only images that don't have a custom tag
                           set by the `image` field.
  -v, --volumes       Remove named volumes declared in the `volumes` section
                       of the Compose file and anonymous volumes
                       attached to containers.
  --remove-orphans    Remove containers for services not defined in the
                       Compose file
```

Docker Compose

docker-compose Befehle

- ▶ docker-compose start / stop / restart / kill / pause / unpause / run
- ▶ docker-compose up / down
- ▶ docker-compose rm

```
Usage: rm [options] [SERVICE...]
```

```
Options:
```

```
-f, --force    Don't ask to confirm removal  
-v            Remove any anonymous volumes attached to containers  
-a, --all     Also remove one-off containers created by  
              docker-compose run
```

tranSMART – Abfrage und Analyse von Daten unterschiedlicher Quellen

- ▶ Open Source Serveranwendung basierend auf i2b2
- ▶ Datawarehouse zur Ablage von Phenotyp- und NGS-Daten
- ▶ Build in: u.a. Genexpressionsdaten, SNP, VCF, mehr durch Community-Erweiterungen
- ▶ Ad-hoc Kohortenerstellung, Export und Analyse von Daten
- ▶ <https://hub.docker.com/u/tmfev/>





cher

anced Workflow SmartR

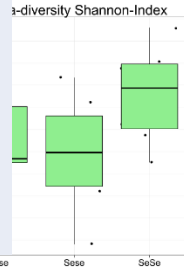
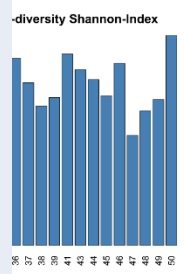
lude X

lude X

lude X

...Mastectom

...Disease Fr



CONTROL

Repositories

Create Repository +

Type to filter repositories by name

	lmfev/2b2 public	2 STARS	41 PULLS	DETAILS
	lmfev/pldgen public	1 STARS	38 PULLS	DETAILS
	lmfev/openclinica public	2 STARS	38 PULLS	DETAILS
	lmfev/ptx public	1 STARS	22 PULLS	DETAILS
	lmfev/gpas public	0 STARS	20 PULLS	DETAILS
	lmfev/glos public	0 STARS	17 PULLS	DETAILS
	lmfev/transmart-db-manage public	0 STARS	17 PULLS	DETAILS
	lmfev/linux-hello-lmf public	0 STARS	17 PULLS	DETAILS

Docker Security Scanning

Protect your repositories from vulnerabilities

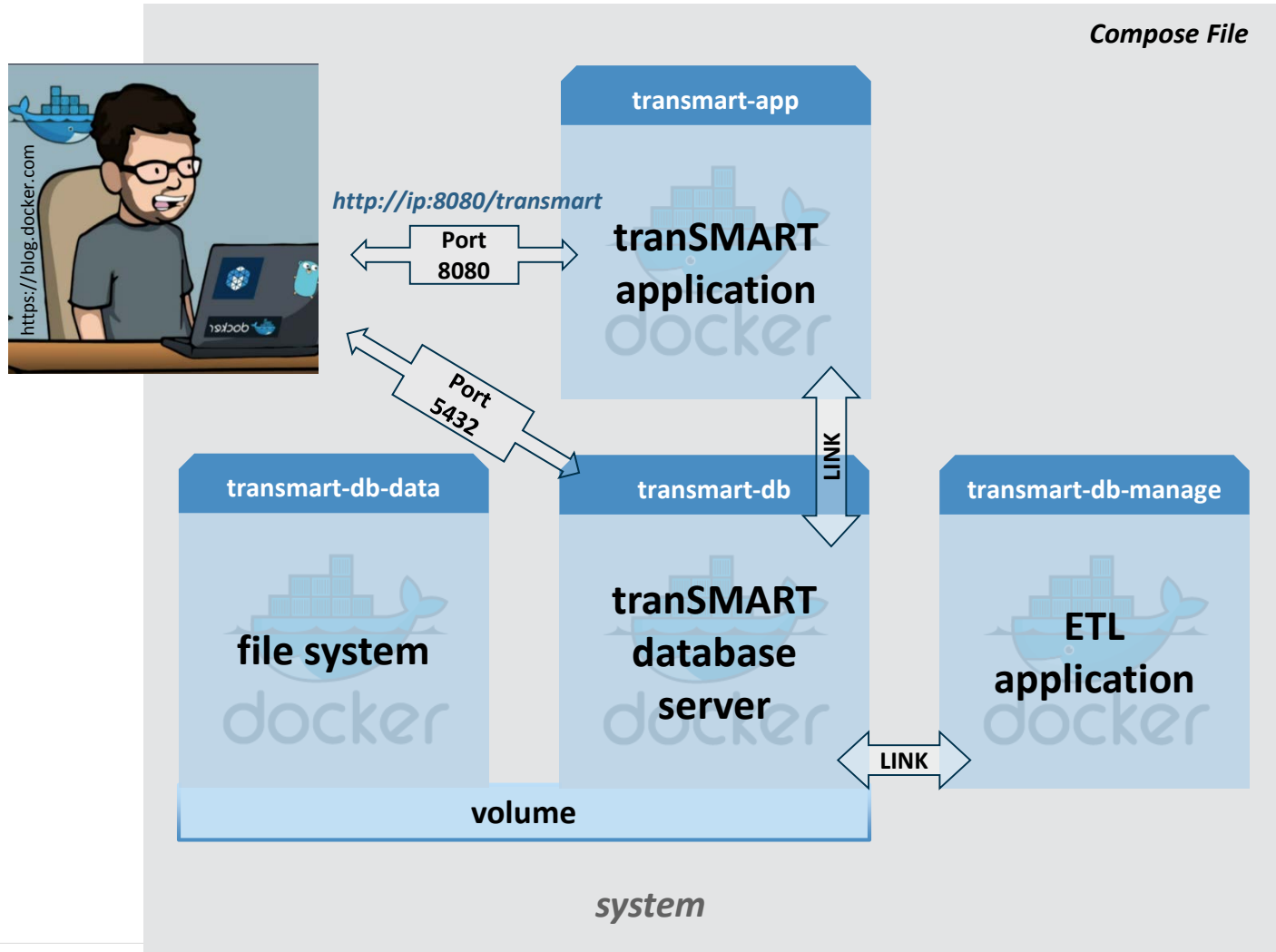
[Try it free](#)

tra

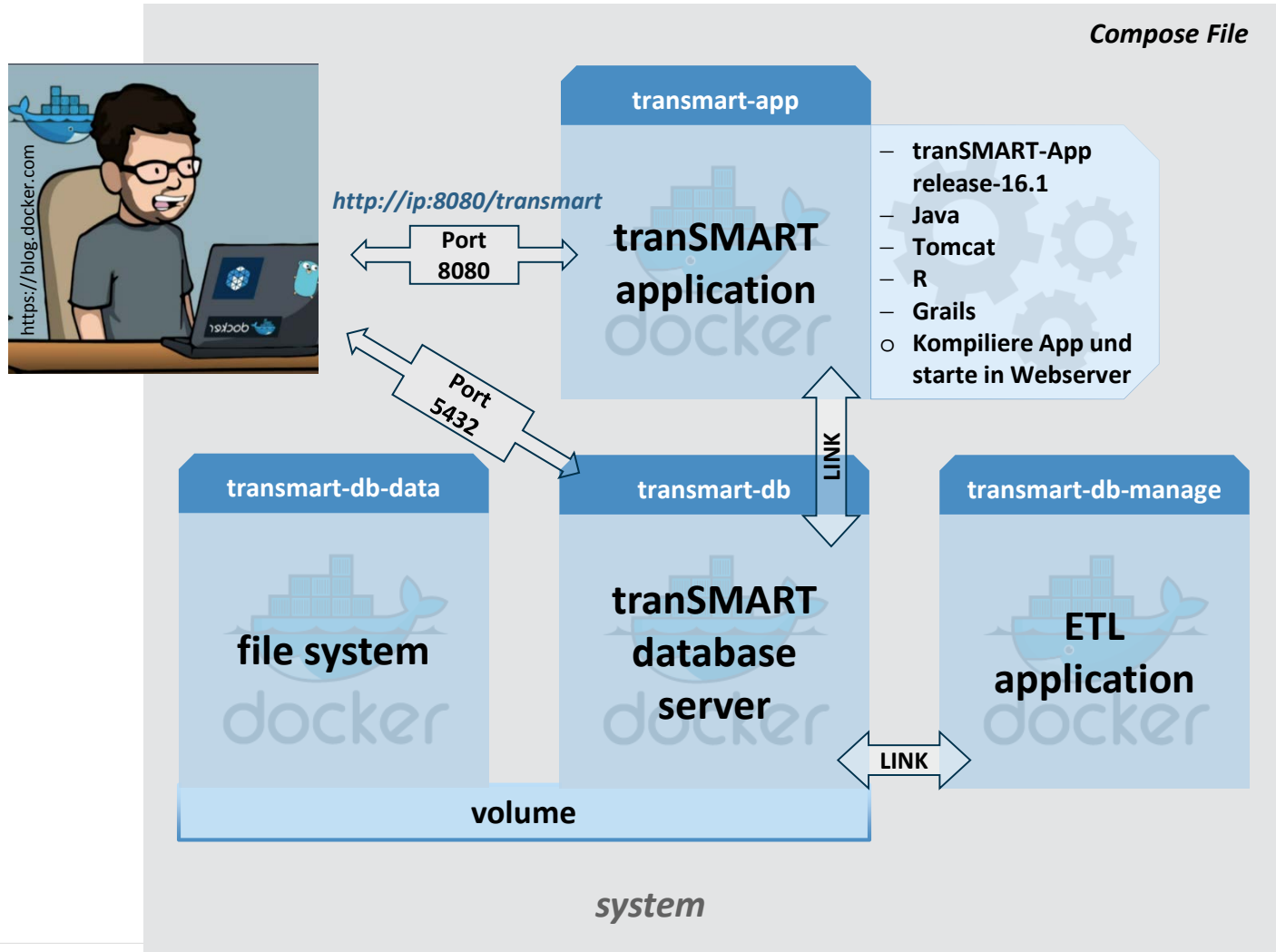
tra
Qu



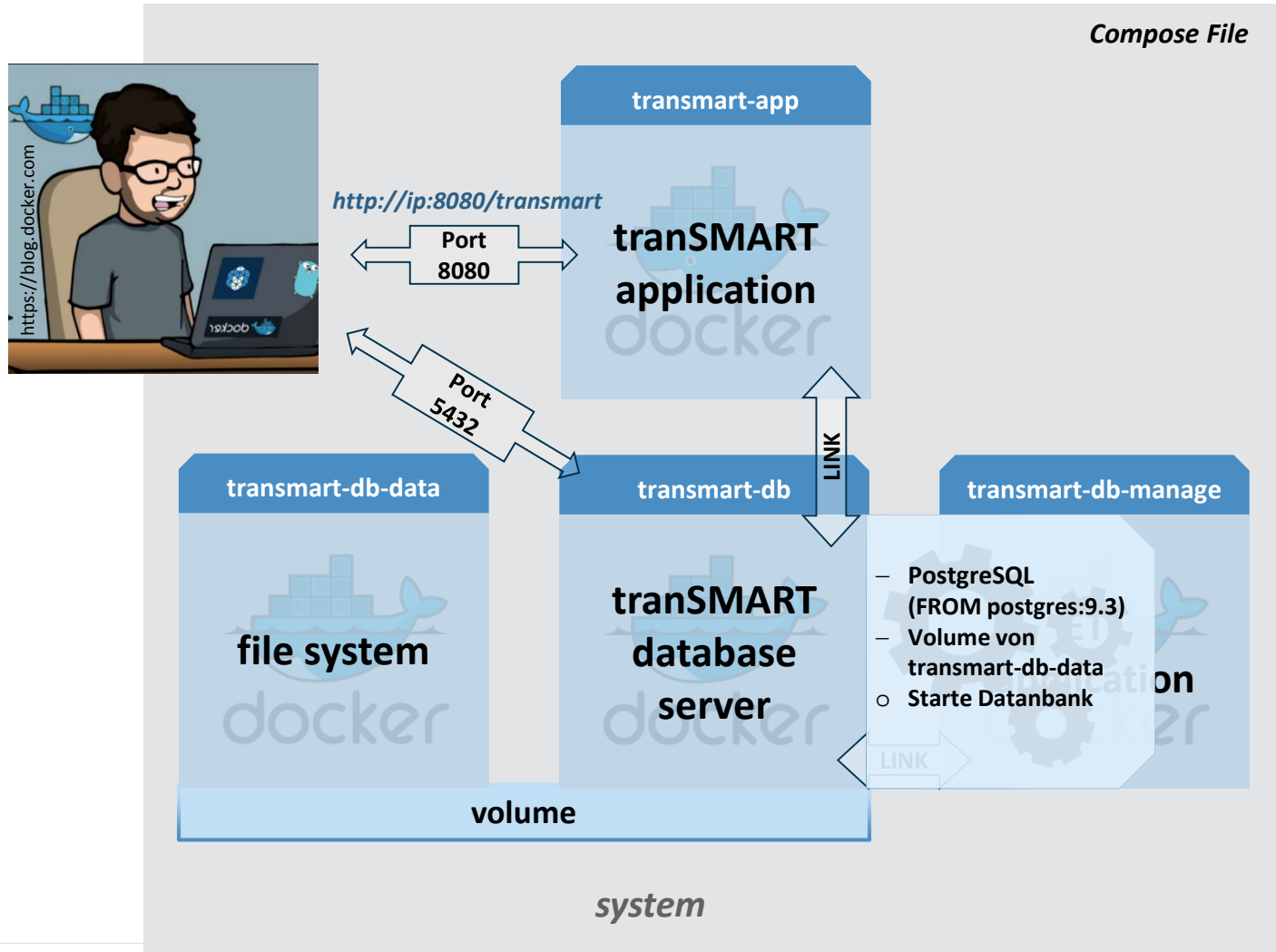
tranSMART Container-Lösung



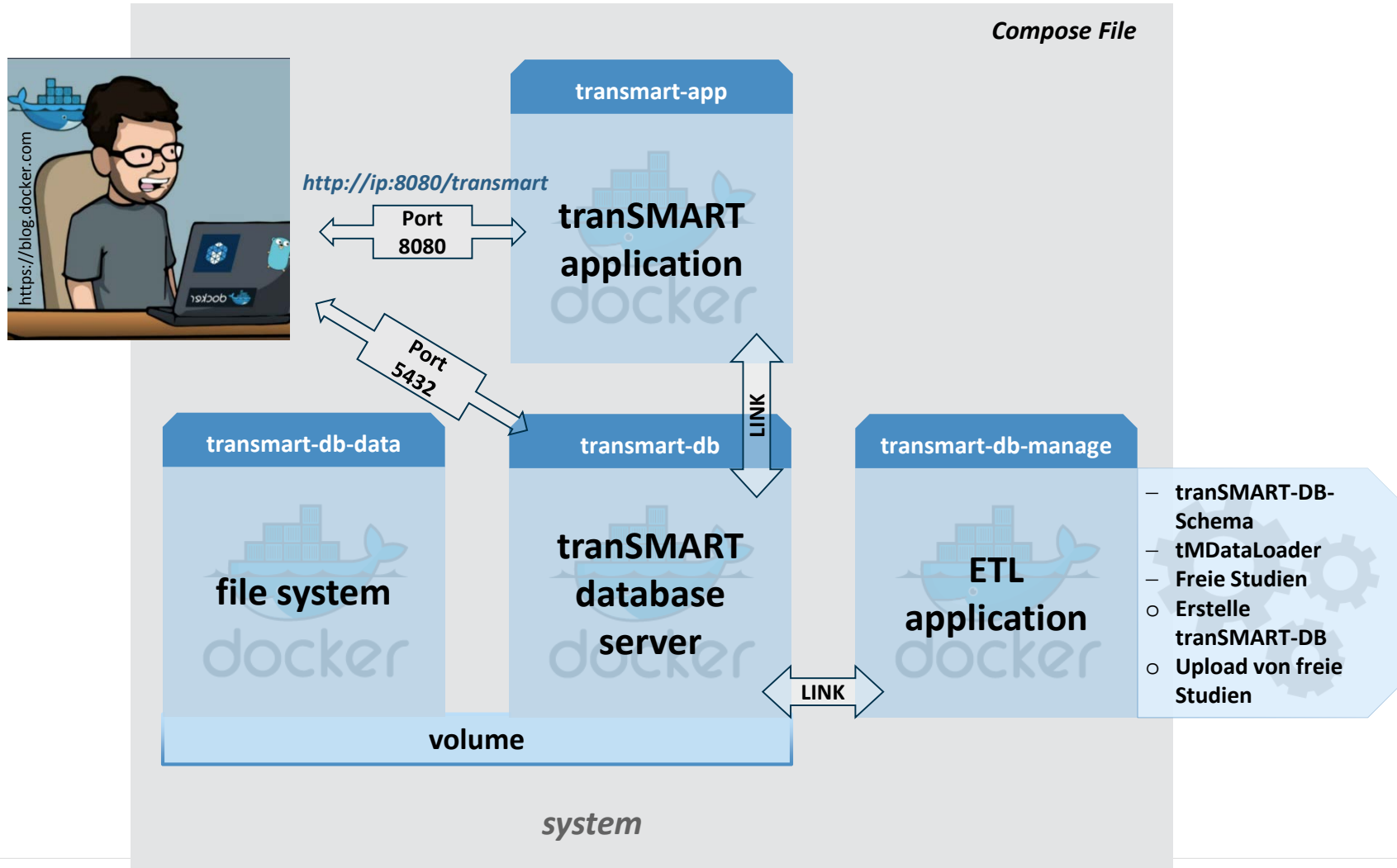
tranSMART Container-Lösung



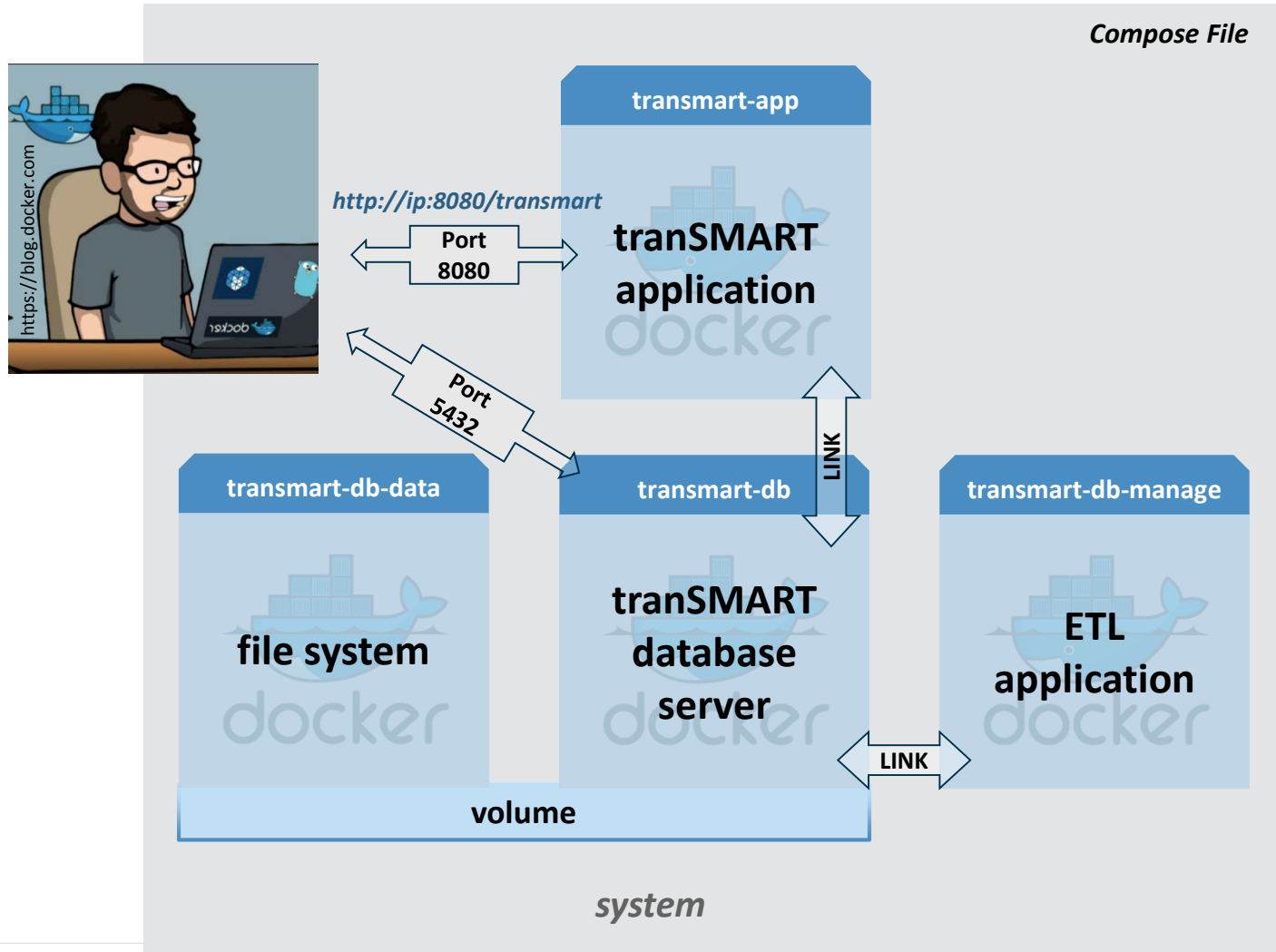
tranSMART Container-Lösung



tranSMART Container-Lösung



tranSMART Container-Lösung



Docker TMF

- Voraussetzungen:
- Oracle VirtualBox
- Docker TMF VM
- PgAdmin3 (Optional) <https://www.pgadmin.org/>
- Putty (Optional) <http://www.putty.org/>
- Webbrowser

TMF Docker VM

- Login: **root**
 - Password: **docker**
 - SSH (putty): **localhost:22**
 - Postgres Login: **postgres**
 - Postgres Password: **docker**
-
- Vorbereitete Übungen unter: **/docker/exercise** Lösungen unter: **/docker/solution**

Praktische Übung

Erstellung eines tranSMART- Docker-Compose

Dockerbank II Praktische Übung: Komplexbeispiel *transSMART*



Linux	root:docker SSH (putty): localhost:22
Postgres	postgres:docker PgAdmin3: localhost:5432 psql -h transmart-db -U <username> -d postgres
Dateien:	Übungen: /docker/exercise Lösungen: /docker/solution Cheatsheet: /docker/dockercheatsheet.txt
transSMART	admin:admin http://localhost:8080/transmart

Nr.	Aufgabe
1	Erstellung einer transSMART-Datenbank
1.1	Verbinden Sie sich mit der laufenden Workshop-VM mit Hilfe von putty (http://www.putty.org) über die Adresse localhost und Port 22. Melden Sie sich mit Benutzernamen/Passwort root/docker an.
1.2	Navigieren Sie in den Ordner <code>cd /docker/exercise/part1</code> .
1.3	Lesen sie die Beschreibungen der zwei im eigenen Docker-Compose zu nutzenden Container https://hub.docker.com/r/tmfev/transmart-db/ sowie https://hub.docker.com/r/tmfev/transmart-db-data/ . Editieren Sie das DockerCompose-File (<code>vim docker-compose.yml</code>), um die bereits eingetragenen Services <code>transmartdbdata</code> und <code>transmartdb</code> zu konfigurieren und zu verbinden.
1.4	Kompilieren und führen Sie Docker-Compose aus: <code>docker-compose up</code>
1.5	Mit Hilfe von PgAdmin3 (https://www.pgadmin.org) oder <code>psql</code> (direkt auf der VM: <code>psql -h transmart-db -U postgres -d postgres</code>) können Sie sich auf der laufenden Instanz von <code>transmart-db</code> auf der Adresse localhost und über den von Ihnen freigegeben Port einloggen. Läuft die Datenbank?
1.6	Stoppen Sie den laufenden Container (<code>strg+c</code>) und löschen Sie ihn anschließend: <code>docker-compose rm -f</code>
2	Installation eines transSMART-Upload-Tools und upload von Beispielstudien
2.1	Navigieren Sie in den Ordner <code>cd /docker/exercise/part2</code> .
2.2	Lesen Sie Beschreibung des Containers https://hub.docker.com/r/tmfev/transmart-db-manage/ . Fügen Sie die fehlenden Einträge für <code>image</code> , <code>depends_on</code> , <code>links</code> und benötigte Variables des <code>environment</code> in YAML-Notation hinzu.
2.3	Kompilieren und führen Sie Ihren Docker-Compose aus: <code>docker-compose up</code>
2.4	Verbinden sie sich mit PgAdmin3 zu Ihrer Datenbank und navigieren Sie zu Datenbanken/transmart/Schemata/i2b2metadata/Tabellen/i2b2 und lassen Sie sich die Daten anzeigen (Rechts-Klick auf „i2b2“->“Daten anzeigen“->“Die obersten 100 Zeilenzeigen“). Psql: <code>psql -h transmart-db -U postgres -d transmart -c „select * from i2b2metadata.i2b2 limit 100;“</code>
2.5	Stoppen Sie den laufenden Container (<code>strg+c</code>) und löschen Sie ihn anschließend: <code>docker-compose rm -f</code>
3	Installation der transSMART-Application
3.1	Navigieren Sie in den Ordner <code>cd /docker/exercise/part3</code> .
3.2	Lesen Sie Beschreibung des Containers https://hub.docker.com/r/tmfev/transmart-app/ . Fügen Sie die fehlenden Einträge für <code>image</code> , <code>depends_on</code> , <code>links</code> , <code>ports</code> und benötigte Variables des <code>environment</code> in YAML-Notation hinzu und ergänzen die neue Option <code>cap_add</code> .
3.3	Kompilieren und führen Sie Ihren Docker-Compose aus: <code>docker-compose up</code>
3.4	Öffnen sie einen Browser und rufen sie die Adresse http://localhost:8080/transmart auf. Loggen sie sich mit admin/admin in Ihre

Dockerbank II Praktische Übung: Komplexbeispiel *transSMART*

) über die Adresse localhost und Port



Linux	root:docker SSH (putty): localhost:22
Postgres	postgres:docker PgAdmin3: localhost:5432 psql -h transmart-db -U <username> -d postgres
Dateien:	Übungen: /docker/exercise Lösungen: /docker/solution Cheatsheet: /docker/dockercheatsheet.txt
transSMART	admin:admin http://localhost:8080/transmart

- 1.4** Kompilieren und führen Sie Docker-Compose aus:
docker-compose up
- 1.5** Mit Hilfe von PgAdmin3 (<https://www.pgadmin.org>) oder psql (direkt auf der VM: psql -h transmart-db -U postgres -d postgres) können Sie sich auf der laufenden Instanz von transmart-db auf der Adresse localhost und über den von Ihnen freigegeben Port einloggen. Läuft die Datenbank?
- 1.6** Stoppen Sie den laufenden Container (strg+c) und löschen Sie ihn anschließend:
docker-compose rm -f
- 2** Installation eines transSMART-Upload-Tools und upload von Beispielstudien
- 2.1** Navigieren Sie in den Ordner cd /docker/exercise/part2.
- 2.2** Lesen Sie Beschreibung des Containers <https://hub.docker.com/r/tmfef/transmart-db-manage/>. Fügen Sie die fehlenden Einträge für image, depends_on, links und benötigte Variables des environment in YAML-Notation hinzu.
- 2.3** Kompilieren und führen Sie Ihren Docker-Compose aus:
docker-compose up
- 2.4** Verbinden sie sich mit PgAdmin3 zu Ihrer Datenbank und navigieren Sie zu Datenbanken/transmart/Schemata/i2b2metadata/Tabellen/i2b2 und lassen Sie sich die Daten anzeigen (Rechts-Klick auf „i2b2“->“Daten anzeigen“->“Die obersten 100 Zeilenzeigen“).
Psql: psql -h transmart-db -U postgres -d transmart -c „select * from i2b2metadata.i2b2 limit 100;“
- 2.5** Stoppen Sie den laufenden Container (strg+c) und löschen Sie ihn anschließend:
docker-compose rm -f
- 3** Installation der transSMART-Application
- 3.1** Navigieren Sie in den Ordner cd /docker/exercise/part3.
- 3.2** Lesen Sie Beschreibung des Containers <https://hub.docker.com/r/tmfef/transmart-app/>. Fügen Sie die fehlenden Einträge für image, depends_on, links, ports und benötigte Variables des environment in YAML-Notation hinzu und ergänzen die neue Option cap_add.
- 3.3** Kompilieren und führen Sie Ihren Docker-Compose aus:
docker-compose up
- 3.4** Öffnen sie einen Browser und rufen sie die Adresse http://localhost:8080/transmart auf. Loggen sie sich mit admin/admin in Ihre transSMART-Instanz ein. In der oberen rechten Ecke können sie über den Link Analyze die geladenen Beispielstudien nutzen.
- 2.5** Stoppen Sie den laufenden Container (strg+c) und löschen Sie ihn anschließend:
docker-compose rm -f
- 4** Lösung
- 4.1** Unter cd /docker/solution befindet sich im Ordner transmartCompose die korrekte Lösung dieser Übung.
- 4.2** Vergleichen Sie die Lösung mit Ihrer.
- 4.3** Kompilieren und führen Sie den TMF transSMART Compose-Container aus:
docker-compose up

Vielen Dank für Ihre Aufmerksamkeit!