# Docker-Compose

- MariaDB mit docker-compose installieren
- Monitoring mit Uptime-Kuma und docker-compose installieren
- Traefik v2 installieren und einrichten
- GLPI mit Docker-Compose installieren

# MariaDB mit docker-compose installieren

## Einleitung

Wenn wir uns mit vielen Daten beschÃ¤ftigen die gespeichert werden sollen, werden relativ schnell Datenbanken verwendet. Datenbanken haben die Eigenschaft das diese sehr effizient mit vielen Daten umgehen kÃ¶nnen. Dazu kommt noch, das mehrere Benutzer gleichzeitig auf die Daten zugreifen kÃ¶nnen. Datenbanken sind in fast jeder Firma und auch in jeglicher Software zu finden.

## Installation

Zuerst mÃ¼ssen wir uns mit unserem Linux Server verbinden. Dies kÃ¶nnen wir Ã¼ber SSH / Telnet oder Ã¼ber eine Serielle Verbindung machen.

Dann mÃ¼ssen wir sicherstellen das **docker** und **docker-compose** auf dem Server installiert ist. Falls nicht, habe ich in diesem Artikel beschrieben wie man die Installation vornimmt: Docker und Docker-Compose installieren

Im nÃ¤chsten Schritt erstellen wir in dem Ordner unser Wahl die **docker-compose.yml** und Ã¶ffnen diese mit dem Editor unserer Wahl und fÃ¼gen dort den folgenden Inhalt ein. Wir verÃ¤ndern daraufhin nur noch das Root Kennwort auf ein entsprechend sicheres Kennwort.

> Ich erstelle fÃ¼r alle Services separate Ordner in denen die Daten der Container gespeichert werden kÃ¶nnen.

```
version: '3.1'

services:
  db:
    image: mariadb:latest
    container_name: db
    restart: always
    environment:
      MARIADB_ROOT_PASSWORD: <passwort>
    volumes:
```

```
    - ./db-data:/var/lib/mysql
  ports:
    - 3306:3306
```

Im nÃ¤chsten Schritt starten wir den Container. Es werden dann die Daten von der Datenbank in dem lokalen Verzeichnis **db-data** angelegt und mit dem Verzeichnis im Container gemountet. Dadurch stellen wir sicher das die Daten auch nach einem Neustart noch vorhanden sind. Im folgenden wird dann auch der *root* Benutzer angelegt und mit dem angegebenen Kennwort versehen.

```
docker-compose up -d
```

Sobald der Container gestartet ist, das erkennen wir an dem done welches erscheint sobald der Container gestartet ist kÃ¶nnen wir uns in den Container hineinschalten um die entsprechenden Datenbanken und Benutzer anzulegen.

```
docker exec -it db /bin/bash
```

Wir sind dann als *root* in dem Container angemeldet und kÃ¶nnen jetzt die MySQL-Shell Ã¶ffnen.

```
mysql -u root -p
```

Und wir geben in der Aufforderung dann unser festgelegtes *root* Kennwort ein. Wir kÃ¶nnen dann nach erfolgter Anmeldung unsere Datenbanken und Benutzer anlegen.

# Datenbank und Benutzer direkt anlegen lassen

Sobald wir eine Datenbank mit einem Benutzer direkt bei der Initialisierung des Containers automatisch anlegen lassen wollen, kÃ¶nnen wir folgende **docker-compose.yml** verwenden. Wir mÃ¼ssen nur den Namen der Datenbank, des Benutzers und das Kennwort des Benutzer entsprechend anpassen.

> Der Benutzer erhÃ¤lt dann alle Berechtigungen nur auf die eine angelegte Datenbank

```
version: '3.1'

services:
  db:
    image: mariadb:latest
    container_name: db
```

```
    restart: always
    environment:
      - MARIADB_ROOT_PASSWORD: <passwort>
      - MARIADB_DATABASE: <datenbank>
      - MARIADB_USER: <benutzer>
      - MARIADB_PASSWORD: <passwort>
    volumes:
      - ./db-data:/var/lib/mysql
    ports:
      - 3306:3306
```

# Monitoring mit Uptime-Kuma und docker-compose installieren

## Einleitung

ZuverlÃ¤ssigkeit ist bei dem Angebot von Server Diensten das A & O. Es gibt Tools die unterstÃ¼tzen uns bei der Ãœberwachung der Dienste. Dazu gehÃ¶rt Uptime-Kuma. Uptime-Kuma ist ein kleines Programm das verschiedene Dienste auf deren Erreichbarkeit Ã¼berprÃ¼ft.

Der Sinn von Ãœberwachungslösungen ist der, das man informiert wird sobald etwas ausfÃ¤llt. Und diese Aufgabe kann Uptime-Kuma auch Ã¼bernehmen. Uptime-Kuma hat ein paar Integrationen mit denen man bei AusfÃ¤llen informiert werden kann.

## Installation

Im ersten Schritt mÃ¼ssen wir uns mit unserem Server per SSH / Telnet / Serielles Kabel verbinden. Im nÃ¤chsten Schritt stellen wir sicher das **docker** und **docker-compose** installiert sind.

Jetzt wechseln wir in einen Ordner unser Wahl und erstellen eine Datei **docker-compose.yml** und Ã¶ffnen diese mit dem Editor unser Wahl und fÃ¼gen dort folgenden Inhalt ein. Wir kÃ¶nnen dort noch den Port Ã¤ndern mit dem wir das Web Interface Ã¶ffnen kÃ¶nnen. Dazu mÃ¼ssen wir dann unter **Ports:** die Zahl nach dem **-** und vor dem **:** Ã¤ndern.

```
version: '3.1'

services:
  uptimekuma:
    image: louislam/uptime-kuma:latest
    restart: always
    container_name: uptime-kuma
    volumes:
      - ./kuma-data:/app/data
```

```
    ports:
      - 3001:3001
```

Als nÃ¤chstes starten wir den Container und initialisieren damit die Anwendung.

```
docker-compose up -d
```

Wenn wir jetzt einige Zeit warten kÃ¶nnen wir die IP-Adresse unseres Servers mit dem entsprechenden Port im Browser angeben und legen uns dann ein Administrator Konto an. Wir kÃ¶nnen dort dann jetzt unsere Services hinzufÃ¼gen die wir Ã¼berwachen wollen.

# Traefik v2 installieren und einrichten

## Einleitung

In dieser Anleitung installieren wir Traefik v2 als Reverse Proxy / Edge Router. Der entscheidende Vorteil von einem Reverse Proxy ist der, das dadurch mehrere Container mit Port **80** nach auÃŸen kommunizieren. So mÃ¼ssen wir uns keine Ports mehr merken, oder Portfreigaben erstellen. Traefik generiert obendrauf noch SSL Zertifikate und erneuert diese automatisch.

Das Routing zwischen den Containern wird Ã¼ber die Subdomain bewerkstelligt. Dadurch weiÃŸ Traefik in welchen Container der Benutzer gelangen soll. Ein Load Balancing lÃ¤sst sich auch einrichten.

Die Installation erfolgt mit docker-compose und dem offiziellen Image von Traefik.

## Traefik Installation und Konfiguration

### Installation der Pakete

Im ersten Schritt mÃ¼ssen wir uns mit unserem Server verbinden damit wir Konsolenzugriff haben. Wichtig dabei ist das auf deinem Linux Server **Docker** und **Docker-Compose** installiert ist. Falls das nicht erledigt ist, wird hier beschrieben wie wir Docker und Docker-Compose installieren:

Docker und Docker-Compose installieren.

Im nÃ¤chsten Schritt mÃ¼ssen wir das Paket **apache2-utils** installieren, da wir das Tool **htpasswd** benÃ¶tigen um spÃ¤ter ein Kennwort fÃ¼r den Administrationsbenutzer zu generieren.

```
sudo apt-get update
sudo apt-get install apache2-utils -y
```

### Dateien und Verzeichnisse anlegen

In diesem Schritt legen wir die benÃ¶tigen Verzeichnisse und Dateien an damit wir Traefik zum laufen zu bekommen. Ich erstelle immer einen zentralen Ordner in dem ich die docker-compose Dateien nach Apps in Ordnern ablege. So kann ich schnell die Container neu starten oder um Konfigurationen vorzunehmen.

```
sudo mkdir /_docker
sudo mkdir /_docker/traefik
sudo mkdir -p /_docker/traefik/data
sudo touch /_docker/traefik/data/acme.json
sudo chmod 600 /_docker/traefik/data/acme.json
sudo touch /_docker/traefik/data/traefik.yml
```

Als zweiten Schritt mÃ¼ssen wir die **traefik.yml** etwas anpassen. Dazu Ã¶ffnen wir die Datei mit einem Editor unserer Wahl. Ich verwende dafÃ¼r den Editor **nano**.

```
sudo nano /_docker/traefik/data/traefik.yml
```

In der Datei fÃ¼gen wir folgenden Inhalt ein:

```
api:
  dashboard: true
entryPoints:
  http:
    address: ":80"
  https:
    address: ":443"
providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    exposedByDefault: false
  file:
    filename: "./dynamic_conf.yml"
certificatesResolvers:
  http:
    acme:
      email: <empfÃ¤nger>@<domain>
      storage: acme.json
      httpChallenge:
        entryPoint: http
```

In der Datei mÃ¼ssen wir unsere E-Mail Adresse eingeben. Diese wird verwendet um euch bei eventuellen Problemen zu benachrichtigen.

Im nÃ¤chsten Schritt erstellen wir die **docker-compose.yml** und fÃ¼gen dort den Inhalt ein.

```
sudo nano /_docker/traefik/docker-compose.yml
```

```
version: '3'

networks:
  traefik:
    external: true

services:
  traefik:
    image: traefik:latest
    container_name: traefik
    restart: always
    security_opt:
      - no-new-privileges:true
    networks:
      - traefik
    ports:
      - 80:80
      - 443:443
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./data/traefik.yml:/traefik.yml:ro
      - ./data/acme.json:/acme.json
      - ./data/dynamic_conf.yml:/dynamic_conf.yml
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.traefik.entrypoints=http"
      - "traefik.http.routers.traefik.rule=Host(`<subdomain>.<domain>`)"
      - "traefik.http.middlewares.traefik-auth.basicauth.users=ADMIN:PASSWORT"
      - "traefik.http.middlewares.traefik-https-redirect.redirectscheme.scheme=https"
      - "traefik.http.routers.traefik.middlewares=traefik-https-redirect"
      - "traefik.http.routers.traefik-secure.entrypoints=https"
      - "traefik.http.routers.traefik-secure.rule=Host(`<subdomain>.<domain>`)"
      - "traefik.http.routers.traefik-secure.tls=true"
      - "traefik.http.routers.traefik-secure.tls.certresolver=http"
      - "traefik.http.routers.traefik-secure.service=<api>@<internal>"
      - "providers.file.filename=/dynamic_conf.yml"
      - "traefik.http.routers.traefik-secure.middlewares=<secHeaders>@<file>,traefik-auth"
```

> Wir mÃ¼ssen in Zeile **36** und **38** eben dann die Eckigen Klammern (<>) kurz entfernen damit die Anwendung funktioniert.

Im nÃ¤chsten Schritt generieren wir das Kennwort um uns in das Webinterface einzuloggen. Dazu Ã¼berlegen wir uns einen Benutzernamen und ein Kennwort. Dieses geben wir dann in den folgenden Befehl ein.

```
echo $(htpasswd -nb <benutzer> <passwort>) | sed -e s/\\$/\\$\\$/g
```

Wir erhalten dann eine Zeichenkette mit dem Benutzernamen und dem verschlÃ¼sselten Kennwort. Diese Informationen fÃ¼gen wir in der **docker-compose.yml** in der 29. Zeile am Ende an. Dazu entfernen wir die Platzhalter *ADMIN:PASSWORT* und tragen dort die Ausgabe des Befehls ein.

Im letzte Schritt legen wir jetzt die **dynamic_conf.yml** an. Dazu geben wir folgenden Befehl ein.

```
sudo nano /_docker/traefik/data/dynamic_conf.yml
```

In der Datei fÃ¼gen wir folgenden Inhalt ein.

```
tls:
  options:
    default:
      minVersion: VersionTLS12
      cipherSuites:
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
        - TLS_AES_128_GCM_SHA256
        - TLS_AES_256_GCM_SHA384
        - TLS_CHACHA20_POLY1305_SHA256
      curvePreferences:
        - CurveP521
        - CurveP384
      sniStrict: true
http:
  middlewares:
    secHeaders:
      headers:
        browserXssFilter: true
        contentTypeNosniff: true
```

```
    frameDeny: true
    sslRedirect: true
    #HSTS Configuration
    stsIncludeSubdomains: true
    stsPreload: true
    stsSeconds: 31536000
    customFrameOptionsValue: "SAMEORIGIN"
```

## Docker Netzwerk anlegen

Jetzt legen wir das Docker Netzwerk an. Das Docker Netzwerk hat die Funktion das die Container die Zugriff auf das Netzwerk haben, darÃ¼ber miteinander kommunizieren. Traefik schiebt den Traefik in die Container Ã¼ber dieses Netzwerk.

```
sudo docker network create traefik
```

## Docker Container starten

Als letztes mÃ¼ssen wir unseren Container starten, dazu geben wir folgenden Befehl ein.

```
docker-compose -f /_docker/traefik/docker-compose.yml up -d
```

Wenn wir jetzt die angegebene Domain in unseren Browser eingeben, erhalten wir ein Anmeldefenster. Wenn wir dort unsere vorhin angegebene Login Daten eingeben, gelangen wir auf das Dashboard von Traefik. Dort kÃ¶nnen wir Fehler und andere Informationen einsehen.

> Wenn wir neue Services zu Traefik hinzufÃ¼gen wollen mÃ¼ssen entsprechende **DNS-EintrÃ¤ge** auf dem **DNS-Server** hinterlegt sein.

# GLPI mit Docker-Compose installieren

## Einleitung

Wir installieren in dieser Anleitung eine Instanz von GLPI. GLPI ist ein Open Source Service Management Tool. In dem Tool kÃ¶nnen GerÃ¤te und Lizenzen Inventarisiert werden, ein StÃ¼ck Dinge dokumentiert werden. GLPI unterstÃ¼tzt dazu noch die Funktion ein Ticket System zu sein. Daher bietet sich das System zur schnellen Installation gut an.

Ich habe dafÃ¼r ein Docker Image erstellt welches wir dafÃ¼r verwenden werden. Das Dockerfile dazu finden wir in meinem Github Repository (Github Repository). So kÃ¶nnen wir nachvollziehen was wir auf unserem Server installieren. Es gibt aber keine Sicherheitsprobleme mit dem Image. Das Image ist minimal gefasst und es wurde nur das notwendigste installiert.

## Installation

### Docker und Docker-Compose installieren

```
volumes:
  glpiServer:
    external: true

networks:
  glpi:
    external: true

services:
  glpi:
    container_name: GLPI-Server
    image: phillipunzen/glpi:latest
    ports:
      - <Port>:80
    networks:
      - glpi
    volumes:
```

```
    - glpiServer:/var/www/html
```

Im ersten Schritt mÃ¼ssen wir Docker und Docker-Compose installieren. Wie wir dies installieren, kÃ¶nnen wir hier nachlesen.

## Docker-Compose.yml anlegen

Im zweiten Schritt navigieren wir in das Verzeichnis in dem wir unsere Konfigurationsdatei ablegen mÃ¶chten und erstellen dann eine neue Datei mit dem Namen **docker-compose.yml** und Ã¶ffnen diese mit einem Editor unserer Wahl.

```
sudo nano docker-compose.yml
```

Dort fÃ¼gen wir folgenden Inhalt ein:

```
volumes:
  glpi:
    external: true

services:
  glpi:
    container_name: GLPI-Server
    image: phillipunzen/glpi-server:10.0.0
    ports:
      - <Port>:80
    volumes:
      - glpi:/var/www/html
```

In der Konfigurationsdatei mÃ¼ssen wir dann noch den Port Ã¤ndern auf dem wir das Webinterface erreichen wollen. Diesen kÃ¶nnen wir frei wÃ¤hlen. Dieser muss aber noch frei sein und nicht von einem Programm verwendet werden.

## Docker Volume anlegen

Im dritten Schritt legen wir jetzt das Volume an mit dem unser Server die Daten fÃ¼r den Webserver ablegt.

```
docker volume create glpi
```

Im nÃ¤chsten Schritt mÃ¼ssen wir eine Datenbank und einen Benutzer anlegen. Diese kÃ¶nnen wir frei wÃ¤hlen. Wie wir einen Benutzer und Datenbank anlegen, kÃ¶nnen wir hier nachlesen.

Wenn wir unsere Datenbank in einem Docker Container haben mÃ¼ssen wir ggf. ein Docker Netzwerk erstellen um die Datenbank zu erreichen. Ãœberprüfe dazu am besten spÃ¤ter die Verbindung zwischen den Containern.

## Container starten

Im vierten Schritt mÃ¼ssen wir unseren Container starten, dies machen wir indem wir folgenden Befehl absetzen:

```
docker-compose up -d
```

## GLPI installieren

Jetzt kÃ¶nnen wir einen Browser Ã¶ffnen und uns mit der Eingabe der IP-Adresse und des Ports auf die Anwendung schalten.

Jetzt installieren wir die Anwendung. Wir klicken uns durch die MenÃ¼s und geben zwischenzeitig die Datenbank Informationen an und gelangen dann schlussendlich zur fertigen Installation von GLPI.

Wenn die Anwendung installiert ist, mÃ¼ssen wir noch die Datei **install.php** lÃ¶schen. Dazu setzen wir folgenden Befehl ab:

```
docker exec -it <Container-Name / ID> rm /var/www/html/install/install.php
```

Jetzt melden wir uns mit den Login Daten im Webinterface an.
**Benutzername:** glpi
**Kennwort:** glpi

> **WICHTIG:** Es wird dringend empfohlen das Kennwort in ein sicheres zu Ã¤ndern!