

Docker

- Docker und Docker-Compose installieren
- 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
- Dockerfile
 - Docker Image aus einem Dockerfile erstellen
 - Docker Container im Image am laufen halten
- Traefik v2
 - Neuen Service zu Traefik hinzufügen
 - Traefik v2 als Reverse Proxy installieren
- Fehler / Probleme
 - Docker-Compose Fehler line 1: Not: command not found
- Docker
 - Alle laufenden Container neustarten

Docker und Docker-Compose installieren

Einleitung

Mit Docker kannst du Anwendungen Containerisieren. Dies bedeutet, dass du auf einem Server auf Anwendungsebene Prozesse voneinander trennen kannst. So kannst du steuern, wer mit wem kommunizieren darf oder ob diese überhaupt von außen erreichbar sein dürfen. So kannst du z.B. mehrere Web-Server auf einem Server installieren.

Installation von Docker

Als Erstes installieren wir Docker. Dies brauchen wir, um erstmal Container zu starten.

Im ersten Schritt aktualisieren wir die Paketquellen und installieren notwendige Pakete

```
sudo apt-get update && \  
sudo apt-get install \  
  ca-certificates \  
  curl \  
  gnupg \  
  lsb-release -y
```

Im zweiten Schritt fügen wir den offiziellen Docker GPG Key hinzu.

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-  
archive-keyring.gpg
```

Im dritten Schritt setzen wir das benötigte Repository auf die **Stable** Version.

```
echo \  
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]  
  https://download.docker.com/linux/debian \  
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Im letzten Schritt aktualisieren wir wieder die Paketquellen und installieren die Docker Pakete.

```
sudo apt-get update && \
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Jetzt können wir überprüfen ob die Docker Installation erfolgreich war

```
sudo docker -v
```

Installation von Docker-Compose

Jetzt installieren wir Docker-Compose. Mit Docker-Compose können wir vorab Konfigurationsdateien für Docker Container erstellen, aus denen wir dann starten. Wir können dann auch ganze Applikationen zu einem Stack zusammenfassen die quasi als eine Einheit gestartet und gestoppt werden.

Im ersten Schritt laden wir das Skript herunter und verschieben es in das Docker Verzeichnis.

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Als Nächstes setzen wir die Berechtigung, um die Befehle ausführen zu können

```
sudo chmod +x /usr/local/bin/docker-compose
```

Falls du dann keine docker-compose Befehle absetzen kannst, führe den unten stehenden Code aus

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Schnellinstallation

Falls wir **Docker** und **Docker-Compose** schnell installieren möchten, können wir den folgenden Befehl eingeben. Dann wird Docker komplett einmal installiert.

```
apt update && apt upgrade -y && apt install sudo -y
apt-get install ca-certificates curl gnupg -y
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture)] signed-by=/etc/apt/keyrings/docker.gpg
https://download.docker.com/linux/debian $(. /etc/os-release && echo "$VERSION_CODENAME)" stable" | sudo
tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

Docker-Compose

MariaDB mit docker-compose installieren

Einleitung

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

Installation

Zuerst müssen wir uns mit unserem Linux Server verbinden. Dies können wir über SSH / Telnet oder über eine Serielle Verbindung machen.

Dann müssen 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 nächsten Schritt erstellen wir in dem Ordner unserer Wahl die **docker-compose.yml** und öffnen diese mit dem Editor unserer Wahl und fügen dort den folgenden Inhalt ein. Wir verändern daraufhin nur noch das Root Kennwort auf ein entsprechend sicheres Kennwort.

Ich erstelle für alle Services separate Ordner in denen die Daten der Container gespeichert werden können.

```
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 nächsten 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 dass 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 können 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 können jetzt die MySQL-Shell öffnen.

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

Und wir geben in der Aufforderung dann unser festgelegtes *root* Kennwort ein. Wir können 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, können wir folgende **docker-compose.yml** verwenden. Wir müssen nur den Namen der Datenbank, des Benutzers und das Kennwort des Benutzer entsprechend anpassen.

Der Benutzer erhält 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

Zuverlässigkeit ist bei dem Angebot von Server Diensten das A & O. Es gibt Tools die unterstützen uns bei der Überwachung der Dienste. Dazu gehört Uptime-Kuma. Uptime-Kuma ist ein kleines Programm das verschiedene Dienste auf deren Erreichbarkeit überprüft.

Der Sinn von Überwachungs Lösungen ist der, das man informiert wird sobald etwas ausfällt. Und diese Aufgabe kann Uptime-Kuma auch übernehmen. Uptime-Kuma hat ein paar Integrationen mit denen man bei Ausfällen informiert werden kann.

Installation

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

Jetzt wechseln wir in einen Ordner unserer Wahl und erstellen eine Datei **docker-compose.yml** und öffnen diese mit dem Editor unserer Wahl und fügen dort folgenden Inhalt ein. Wir können dort noch den Port ändern mit dem wir das Web Interface öffnen können. Dazu müssen wir dann unter **Ports:** die Zahl nach dem - und vor dem : ändern.

```
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 nächstes starten wir den Container und initialisieren damit die Anwendung.

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

Wenn wir jetzt einige Zeit warten können wir die IP-Adresse unseres Servers mit dem entsprechenden Port im Browser angeben und legen uns dann ein Administrator Konto an. Wir können dort dann jetzt unsere Services hinzufügen die wir überwachen 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, dass dadurch mehrere Container mit Port **80** nach außen kommunizieren. So müssen 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 über die Subdomain bewerkstelligt. Dadurch weiß Traefik in welchen Container der Benutzer gelangen soll. Ein Load Balancing lässt 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 müssen wir uns mit unserem Server verbinden damit wir Konsolenzugriff haben. Wichtig dabei ist dass 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 nächsten Schritt müssen wir das Paket **apache2-utils** installieren, da wir das Tool **htpasswd** benötigen um später ein Kennwort für 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 benötigten 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 müssen wir die **traefik.yml** etwas anpassen. Dazu öffnen wir die Datei mit einem Editor unserer Wahl. Ich verwende dafür den Editor **nano**.

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

In der Datei fügen 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: <empfangen>@<domain>
      storage: acme.json
      httpChallenge:
        entryPoint: http
```

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

Im nächsten Schritt erstellen wir die **docker-compose.yml** und fügen 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:PASSWORD"

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

Im nächsten Schritt generieren wir das Kennwort um uns in das Webinterface einzuloggen. Dazu überlegen 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 verschlüsselten Kennwort. Diese Informationen fügen 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 fügen 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, darüber miteinander kommunizieren. Traefik schiebt den Traefik in die Container über dieses Netzwerk.

```
sudo docker network create traefik
```

Docker Container starten

Als letztes müssen 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 können wir Fehler und andere Informationen einsehen.

Wenn wir neue Services zu Traefik hinzufügen wollen müssen entsprechende **DNS-Einträge** 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 können Geräte und Lizenzen inventarisiert werden, ein Stück Dinge dokumentiert werden. GLPI unterstützt dazu noch die Funktion ein Ticket System zu sein. Daher bietet sich das System zur schnellen Installation gut an.

Ich habe dafür ein Docker Image erstellt welches wir dafür verwenden werden. Das Dockerfile dazu finden wir in meinem Github Repository ([Github Repository](#)). So können 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 müssen wir Docker und Docker-Compose installieren. Wie wir dies installieren, können wir [hier](#) nachlesen.

Docker-Compose.yml anlegen

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

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

Dort fügen 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 müssen wir dann noch den Port ändern auf dem wir das Webinterface erreichen wollen. Diesen können wir frei wählen. 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 für den Webserver ablegt.

```
docker volume create glpi
```

Im nächsten Schritt müssen wir eine Datenbank und einen Benutzer anlegen. Diese können wir frei wählen. Wie wir einen Benutzer und Datenbank anlegen, können wir [hier](#) nachlesen.

Wenn wir unsere Datenbank in einem Docker Container haben müssen wir ggf. ein Docker Netzwerk erstellen um die Datenbank zu erreichen. Überprüfen Sie dazu am besten später die Verbindung zwischen den Containern.

Container starten

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

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

GLPI installieren

Jetzt können wir einen Browser öffnen 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 Menüs und geben zwischenzeitig die Datenbank Informationen an und gelangen dann schlussendlich zur fertigen Installation von GLPI.

Wenn die Anwendung installiert ist, müssen wir noch die Datei **install.php** löschen. 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 ändern!

Dockerfile

Docker Image aus einem Dockerfile erstellen

Einleitung

In diesem Beitrag erkläre ich kurz wie du das einem Dockerfile ein Image erstellen kannst. Dieses kannst du dann weiter publizieren und verwenden. Docker ist kurz gesagt eine Software die sogenannte **Container** verwendet. Mit diesen können abgeschottete Systeme erstellt werden.

Diese teilen sich im Gegensatz zu virtuellen Maschinen den Kernel mit dem Host und bauen ihre Layer auf. In den mehreren Layer befinden sich dann die installierten Programme und Abhängigkeiten.

Docker unterstützt nur einen Linux Kernel. Wenn du Docker auf einem Windows System installierst, lässt Docker eine Linux Maschine auf dem Host laufen. Die Container greifen dann auf den Linux Kernel im Windows System zu.

In Windows:

Wenn du in Windows ein Docker-Image erstellen möchtest, öffnest du die PowerShell als **Administrator**. Nun navigierst du in das Verzeichnis in dem sich das **Dockerfile** befindet.

Nun verwendest du den **docker build** Befehl. Die Syntax dieses Befehls ist folgend aufgebaut:

```
docker build [parameter] .
```

Das Image wird nun erstellt und lokal abgelegt. Du kannst dieses lokal nun verwenden.

Am besten gibst du gleich einen Tag für das Image mit. Dann kannst du das Image leichter wieder finden.

```
docker build -t phillipunzen/apache-php80:latest .
```

In Linux:

Wenn du in Linux ein Docker-Image erstellen möchtest, öffnest du das Terminal. Nun navigierst du in das Verzeichnis in dem sich das **Dockerfile** befindet.

Nun verwendest du den **docker build** Befehl. Die Syntax dieses Befehls ist folgend aufgebaut:

```
docker build [parameter] .
```

Das Image wird nun erstellt und lokal abgelegt.

Am besten gibst du gleich einen Tag für das Image mit. Dann kannst du das Image leichter wieder finden.

docker build -t phillipunzen/apache-php80:latest .

Parameter

Wenn du ein Docker-Image erstellst, kannst du den **Build Prozess** mit Parametern ein bisschen modifizieren. In dieser Tabelle findest du die Parameter die du verwenden kannst.

Docker Build Parameter		
Parameter	Beschreibung	Beispiel
-t	Mit diesem Befehl kannst du ein Image taggen. Dieses kann man als Namen verstehen. Mit diesem Namen wird es dann aufgerufen oder ggf. in Docker Hub heruntergeladen wenn du das dort hochlädst.	docker build -t phillipunzen/apache-php74 .
--label	Mit diesem Parameter kannst du Meta Daten zum Image hinzufügen.	docker build --label LABEL .

Docker Container im Image am laufen halten

Einleitung

Wenn wir eigene **Dockerimages** erstellen, kann es vorkommen, dass der Container nach einmaligen Ausführen sofort beendet wird. Dies kann ärgerlich sein, wenn auf dem Container ein Webserver oder Ähnliches läuft. Damit der Container nach Ausführung des Codes, noch weiter läuft, müssen wir lediglich nur einen Absatz im Dockerfile hinzufügen. Sobald wir das Image dann neu bauen lassen, und dieses dann ausführen, sollte der Container nicht absterben.

Container laufen lassen

Im **Dockerfile** müssen wir nur den nachstehenden Code einfügen.

```
CMD tail -f /dev/null
```

Wenn wir jetzt das **Dockerimage** neu erstellen lassen, mit `docker build` wird beim Ausführen des Containers, der Code im **CMD Teil** ausgeführt. Mit dem Befehl produzieren wir eine Endlosschleife, wodurch unser Container nicht automatisch herunterfährt.

Traefik v2

Neuen Service zu Traefik hinzufügen

Einleitung

Mithilfe von Traefik ist es möglich einen zentralen Proxy Server für alle Web Applikationen zu verwenden. Dieser nimmt alle Anfragen standardgemäß auf Port **80** und **443** an und leitet diese anhand der **Subdomain** in die entsprechenden **Docker Container**. Dies hat den Vorteil das die Applikationen nur über die Domains aufgerufen werden können, und so keine speziellen Ports mehr angegeben werden müssen oder Anwendungen auf andere Ports umgeleitet werden müssen.

Anwendung

Um eine Anwendung hinzuzufügen, musst du Traefik installieren. Wie du das erledigst erfährst du HIER (Folgt noch...)

Nun öffnest du die **docker-compose.yml** und fügst folgende Label ein:

```
labels:
  - "traefik.enable=true"
  - "traefik.docker.network=traefik"
  - "traefik.http.routers.service.entrypoints=http"
  - "traefik.http.routers.service.rule=Host(`domain.de`,`www.domain.de`)"
  - "traefik.http.middlewares.service-https-redirect.redirectscheme.scheme=https"
  - "traefik.http.routers.service.middlewares=service-https-redirect"
  - "traefik.http.routers.service-secure.entrypoints=https"
  - "traefik.http.routers.service-secure.rule=Host(`domain.de`,`www.domain.de`)"
  - "traefik.http.routers.service-secure.tls=true"
  - "traefik.http.routers.service-secure.tls.certresolver=http"
  - "traefik.http.routers.service-secure.service=service"
  - "traefik.http.services.service.loadbalancer.server.port=80"
  - "traefik.http.routers.service-secure.middlewares=secHeaders@file"
  - "com.centurylinklabs.watchtower.enable=true"
```

Dort musst du nur noch die Domains anpassen, den Ziel-Port angeben, das Traefik Netzwerk angeben und dem Service einen Namen geben. Du veränderst dann das Schlüsselwort service

in den entsprechenden App-Namen. Zuletzt startest du den Docker Container, und dein Container ist nun Ã¼ber Traefik erreichbar!

Notiz: Der letzte Label dient dazu Ã¼ber **Watchtower** automatisch **Image Updates** der service zu installieren. Wenn du kein Watchtower verwendest, kannst du dieses entfernen.

Traefik v2 als Reverse Proxy installieren

Einleitung

Traefik ist ein sogenannter **Reverse Proxy**. Das bedeutet, dieser nimmt alle Anfragen auf den angegebenen *Entrypoints* an und routet die Anfragen in die entsprechenden *Docker Container*. Dies hat den Vorteil dass keine *Virtuellen Hosts* oder du dir unterschiedliche *Port-Nummern* merken musst. Deine Anfragen kannst du automatisch auf einen angegebenen Port routen.

Mehr Informationen: https://hub.docker.com/_/traefik/

Installation

Wenn wir nun **Traefik** installieren wollen, musst du **Docker** installiert haben. Wie du das machst, habe ich **hier** erklärt. Im ersten Schritt installieren wir das Tool **htpasswd**.

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

Im nächsten Schritt legen wir ein Verzeichnis für **Traefik** an. In meinem Beispiel haben wir unter dem root Verzeichnis, ein Verzeichnis mit dem Namen **_docker**. Dort liegen alle Daten für die **Docker Container**. In diesem Verzeichnis legen wir einen Ordner für **Traefik** an.

```
sudo mkdir -p /_docker/traefik
```

Im nächsten Schritt erstellen wir ein paar Verzeichnisse und Dateien die zur späteren Konfiguration oder Dateiablage benötigt werden.

```
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
```

im vierten Schritt bearbeiten wir nun die **traefik.yml**. Dies ist eine *Konfigurationsdatei*. Mit dieser kannst du **Traefik** konfigurieren damit es so arbeitet wie du es brauchst. Dort musst du nur die E-Mail Adresse verändern. Da die Zertifikate per **Let's Encrypt** beantragt werden, kannst du damit

deine E-Mail gleich für weitere Rückfragen hinterlegen. Dort fügst du dann folgende Konfiguration ein.

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

```
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: beispiel@beispiel.de # <--- E-Mail Adresse für Rückfragen
      storage: acme.json
      httpChallenge:
        entryPoint: http
```

Im Anschluss erstellen wir die **docker-compose.yml**. Diese Datei ist quasi die Konfigurationsdatei von *Applikationen* die du mit Docker erstellst. Aus dieser Datei erstellst du Anwendungen und gibst die entsprechenden Parameter mit.

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

```
version: '3'
services:
  traefik:
    image: traefik:latest
    container_name: traefik
    restart: unless-stopped
    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(`traefik.beispiel.de`)"
- "traefik.http.middlewares.traefik-auth.basicauth.users=admin:Pa$$w0rd"
- "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(`traefik.beispiel.de`)"
- "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"
networks:
traefik:
external: true

```

Jetzt erstellen wir einen Benutzer und geben dem ein entsprechendes Passwort mit. Es wird **dringend** empfohlen dieses zu Ändern! In der Konfigurationsdatei sind momentan die Platzhalter **admin** und **Pa\$\$w0rd** dort eingetragen. Der Benutzername und das Kennwort werden durch ein : voneinander getrennt. Wir generieren jetzt ein neues Kennwort mit dem Tool **htpasswd**. Dort ersetzt du den Benutzernamen und das Kennwort durch deine eigenen Daten.

```
echo $(htpasswd -nb admin Pa$$w0rd) | sed -e s/\$/\$/g
```

Du bekommst beim ausführen dieses Befehls eine Ausgabe, diese Ausgabe trägst du in das entsprechende Label in der **docker-compose.yml** ein.

Vorher:

```
- "traefik.http.middlewares.traefik-auth.basicauth.users=admin:Pa$$w0rd"
```

Nachher:

```
- "traefik.http.middlewares.traefik-auth.basicauth.users=admin:$$apr1$$ofCridn5$$94xGJcgJ0qoS0tZUHN.gP/"
```

Als nächstes erstellen wir eine weitere **Konfigurationsdatei**. Diese erstellen wir mit dem folgenden Befehl und fügen dann wieder einen Inhalt ein.

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

```
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"
```

Im vorletzten Schritt legen wir das **Docker Netzwerk** an indem Traefik mit den anderen Containern kommuniziert.

```
sudo docker network create traefik
```

Und im letzten Schritt starten wir den Container mit dem **docker-compose** Befehl.

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

Wenn du jetzt deinen Webbrowser öffnest und auf den entsprechenden Hostnamen navigierst, wirst du dich mit den vorhin angegebenen Daten anmelden müssen. Wenn du das gemacht hast, bist du auf dem Traefik Dashboard angemeldet. Dort findest du alle Informationen über Verbindungen, Entrypoints und andere Dienste.

Fehler / Probleme

Docker-Compose Fehler line 1: Not: command not found

Einleitung

Wenn wir auf einem Raspberry Docker und Docker-Compose installieren kann es mal zu folgender Fehlermeldung kommen:

```
/usr/local/bin/docker-compose : line 1: Not: command not found
```

In dieser Anleitung werden wir den Fehler beheben.

Lösung

Um den Fehler zu beheben müssen wir zwei Befehle absetzen.

Wir laden die letzte Version von Docker herunter ohne Parameter anzugeben.

```
curl -L "https://github.com/docker/compose/releases/download/$(curl  
https://github.com/docker/compose/releases | grep -m1 '<a href="/docker/compose/releases/download/' | grep -  
o 'v[0-9:].[0-9].[0-9]')/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Als letzten Schritt, setzen wir nur noch die Berechtigungen auf die Datei. Dann sollte alles wieder laufen :)

```
chmod +x /usr/local/bin/docker-compose
```


Docker

Alle laufenden Container neustarten

Einleitung

In diesem Beitrag erkläre ich kurz, wie wir mit einem Befehl alle laufenden **Docker Container** neu starten können. Dies kann manchmal hilfreich sein, wenn Updates eingespielt werden und der Neustart möglichst schnell durchgeführt werden soll.

Container neu starten

Um die Container neu starten, müssen wir nur einmal den folgenden Befehl ausführen. Es werden beim Ausführen, nach und nach die entsprechenden Container ID's angezeigt, von dem Container welcher gerade neu gestartet wird.

```
docker restart $(docker ps -q)
```