Traefik

Ein Reverse Proxy der sich durch sehr gute Performance und leichter Konfiguration hervorhebt. Besonders einfach ist z. B. das Konfigurieren von Docker Container, um diese über den Proxy abzuschirmen. Außerdem kann Traefik native mit Let's Encrypt umgehen und Zertifikate beziehen.

Installation

Traefik lässt sich bequem als Docker Container bereitstellen. Es wird nur ein Container für Traefik selbst benötigt, keine weiteren Zusatzdienste sind notwendig.

Eine einfache voll funktionsfähige Konfiguration als Docker Compose sieht z. B. wie folgt aus. Diese ist sehr gut für Testumgebungen geeignet, um neue Images auszuprobieren und diese auf einem Host unter einem Port (hier 80) bereitzustellen.

> Für ein produktives Setup sollte unbedingt noch Let's Encrypt mit konfiguriert werden. Auch wenn diese Konfiguration bereits reicht, um Dienste anzubieten, so währen diese jetzt noch unverschlüsselt und somit unsicher. Außerdem ist in diesem Fall mit api.insecure die Konfiguration über Port 8080 einsehbar, was ein Sicherheitsrisiko bedeutet.

```
services:
  traefik:
    image: "traefik"
    container_name: "traefik"
    command:
      #- "--log.level=DEBUG"
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
```

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Damit wäre Traefik nun einsatzbereit und unter http://localhost:8080 die Konfigurationsoberfläche erreichbar.

Reverse Proxy Netzwerk konfigurieren

Um nun Dienste über den Traefik erreichbar zu machen, muss dieser die Dienste innerhalb des Docker Hosts erreichen können. Da Docker für jede Compose Datei ein eigenes Default Netzwerk konfiguriert, kann Traefik diese nicht erreichen. Auch die Option alle Dienste zusammen mit Traefik in einer Compose Datei zu beschreiben, ist keine gute Idee.

Somit erstellen wir uns nun ein Netzwerk, welches unabhängig von den Compose Dateien verwaltet wird. In diesem Beispiel wird das Netzwerk mit dem Namen *traefik* erstellt. Da keine weiteren Optionen angegeben werden, ist es ein Bridge Netzwerk, welches für unseren Fall gut geeignet ist.

```
sudo docker network create traefik
```

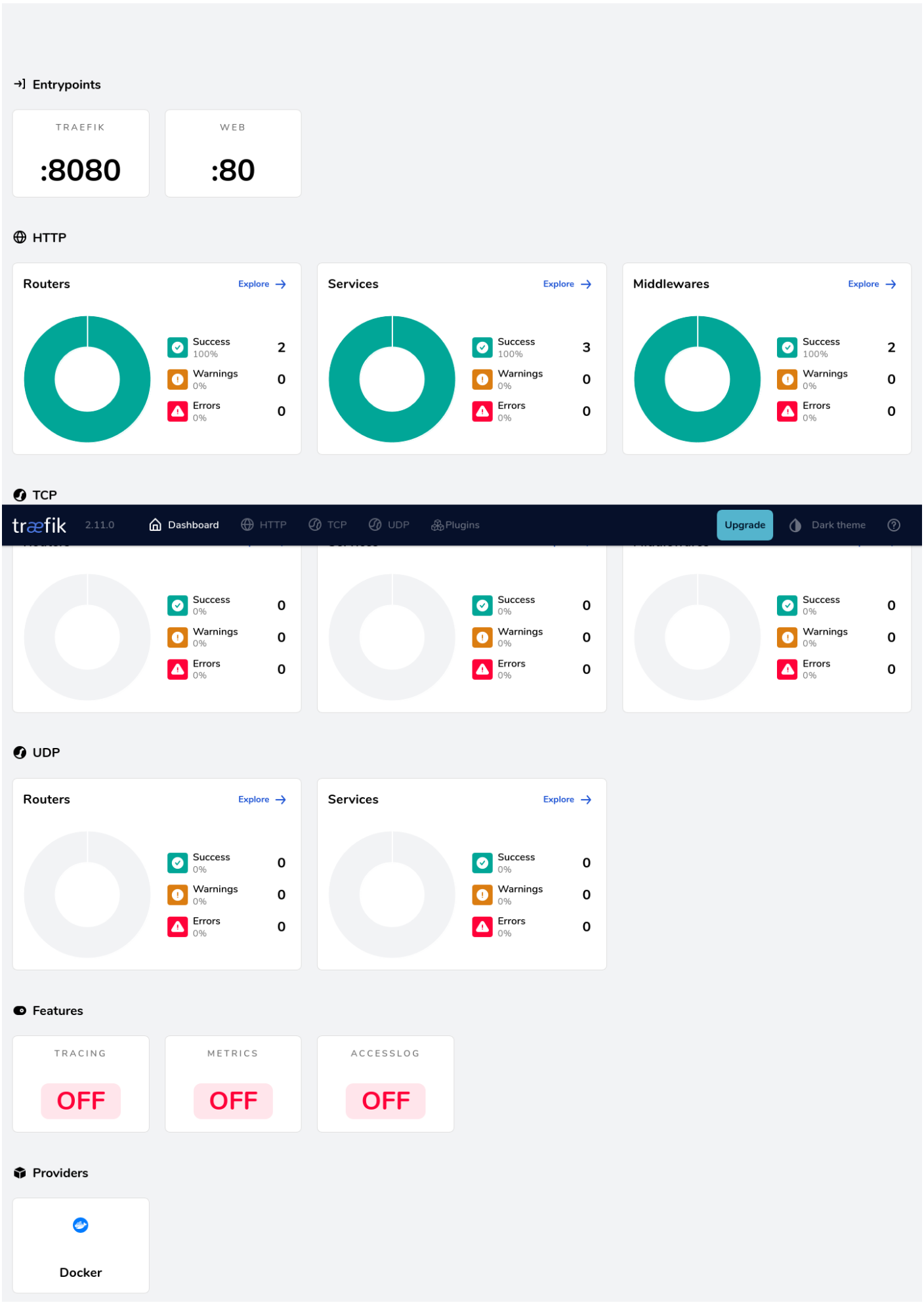
Nun wird die bisherige Compose Konfiguration für den Container Traefik um folgende Zeilen erweitert, damit Traefik in das Netzwerk aufgenommen wird.

```
    command:
      - "--providers.docker.network=traefik"
    networks:
      - traefik
networks:
  traefik:
    name: traefik
    external: true
```

Zum Schluss den Container mit *docker compose up -d* neu erstellen lassen und fertig. Ab jetzt können einfach weitere Container in dieses Netzwerk aufgenommen werden, damit Traefik sie erreichen und anbieten kann.

Konfiguration einsehen

Über den Port 8080 lässt sich die Konfiguration anzeigen: http://localhost:8080. Es wird ein Dashboard angezeigt, welches alle erkannten Dienste, Regeln usw. anzeigt. Hier kann nun erkundet werden, ob alles richtig übernommen wurde. Für produktive Umgebungen, die öffentlich erreichbar sind, sollte dieses Dashboard deaktiviert werden, da die gesamte Konfiguration gezeigt wird und alles offen legt.


Entrypoints
TRAEFIK
:8080
WEB
:80
HTTP
Routers
Explore
Success
100%
2
Warnings
0%
0
Errors
0%
0
Services
Explore
Success
100%
3
Warnings
0%
0
Errors
0%
0
Middlewares
Explore
Success
100%
2
Warnings
0%
0
Errors
0%
0
TCP
traefik
2.11.0
Dashboard
HTTP
TCP
UDP
Plugins
Upgrade
Dark theme
Success
0%
0
Warnings
0%
0
Errors
0%
0
Success
0%
0
Warnings
0%
0
Errors
0%
0
Success
0%
0
Warnings
0%
0
Errors
0%
0
UDP
Routers
Explore
Success
0%
0
Warnings
0%
0
Errors
0%
0
Services
Explore
Success
0%
0
Warnings
0%
0
Errors
0%
0
Features
TRACING
OFF
METRICS
OFF
ACCESSLOG
OFF
Providers
Docker

HTTPS aktivieren

Die bisherige Konfiguration von Traefik ist noch unsicher, da keine Verschlüsselung verwendet wird. Diese lässt sich jedoch nachrüsten, indem die Konfiguration angepasst wird. Traefik kann von Haus aus mit Let's Encrypt umgehen, was die Verwaltung von Zertifikaten recht einfach macht. Es sind keine zusätzlichen Tools notwendig.

> Bei der Konfiguration der Challenge für die Zertifikate wird hinter certificatesresolvers der Name festgelegt. Im folgenden Beispiel wird der Name des Resolvers auf *einzelzertifikat* festgelegt, indem dieser einfach mit einem Punkt getrennt angehängt wird. Der Name ist wichtig, denn dieser wird später in allen Containern als Label angegeben werden müssen, die mit Zertifikaten von diesem Resolver versorgt werden sollen.

```
services:
  traefik:
    image: traefik
    container_name: traefik
    restart: unless-stopped
    ports:
      - 80:80
      - 443:443
      - 8080:8080
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ${TRAEFIK_PFAD}/traefik_data:/traefik
    command:
      - --api.dashboard=true
      - --api.insecure=true
      - --providers.docker=true
      - --providers.docker.exposedbydefault=false
      - --providers.docker.network=traefik
      - --entrypoints.webinsecure.address=:80
      - --entrypoints.websecure.address=:443
      - --entrypoints.traefik.address=:8080
# Challenge für die Zertifikate festlegen
      - --certificatesresolvers.einzelzertifikat.acme.tlschallenge=true
      - --certificatesresolvers.einzelzertifikat.acme.email=${ACME_EMAIL}
```

```
      - --
certificatesresolvers.einzelzertifikat.acme.storage=/traefik/certs_einzelzertifikat/acme.json
    labels:
      - traefik.enable=true
      # HTTPS Umleitung
      - traefik.http.routers.http-catchall.entrypoints=webinsecure
      - traefik.http.routers.http-catchall.rule=HostRegexp(`{host:.+}`)
      - traefik.http.routers.http-catchall.middlewares=redirect-to-https
      - traefik.http.middlewares.redirect-to-https.redirectscheme.scheme=https
      # HTTP Router
      - traefik.http.routers.traefik-secure.entrypoints=websecure
      - traefik.http.routers.traefik-secure.tls=true
      - traefik.http.routers.traefik-secure.tls.domains[0].main=*.${DOMAIN}
      - traefik.http.routers.traefik-secure.tls.domains[0].sans=${DOMAIN}
      - traefik.http.routers.traefik-secure.service=api@internal
    networks:
      - traefik

networks:
  traefik:
    name: traefik
    external: true
```

> In der oben gezeigten Konfiguration ist noch das Dashboard verfügbar, welches alle Interna von Traefik anzeigt. Für ein produktives Setup, dass von außen erreichbar ist, sollte dieses besser abgeschaltet werden. Dazu die folgenden Zeilen löschen:
>
> - 8080:8080
> - --entrypoints.traefik.address=:8080
> - traefik.http.routers.traefik-secure.rule=Host(`traefik.${DOMAIN}`)

In dieser Compose Konfiguration werden Variablen verwendet, um die Einstellung zu vereinfachen. Nachdem die docker-compose.yml erstellt wurde, gilt es noch folgende .env Datei zu erstellen und anzupassen.

```
TRAEFIK_PFAD=/pfad/zu/traefik
ACME_EMAIL=email@beispiel.de
DOMAIN=beispiel.de
```

Bevor nun die Konfiguration angewendet wird, müssen schon der Ordner traefik_data und darin der Unterordner für die Zertifikate erstellt werden. Zusätzlich ist die Datei für die Zertifikatskonfiguration zu erstellen und die Berechtigung anzupassen. Dazu kann das folgende

Skript verwendet werden, wobei der Pfad anzupassen ist.

```
pfad=/pfad/zu/traefik/traefik_data/certs_einzelzertifikat
mkdir $pfad
touch mkdir $pfad/acme.json
chmod 600 $pfad/acme.json
```

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Anschließend müssen nur noch die gewünschten Dienst konfiguriert werden, damit diese von Traefik verwaltet werden können.

> In der Konfiguration ist noch die Konfigurationsübersicht unter Port 8080 erreichbar. Diese sollte in einer produktiven Umgebung deaktiviert werden.

HTTPS mit Wildcard aktivieren

Auch ein Wildcard Zertifikat kann Traefik von Let's Encrypt beantragen indem es die DNS-Challenge durchführt. Dafür sind ebenfalls keine zusätzliche Tools notwendig.

> Die DNS-Challenge setzt voraus, dass der Domänenanbieter diese auch unterstützt. Bei GoDaddy ist dies der Fall, weswegen die folgenden Konfigurationen für diesen Anbieter durchgeführt werden. Für die Challenge muss ein API Key samt Secret generiert werden.

Für die DNS-Challenge sind mehr Optionen zu definieren, da ein DNS-Eintrag erstellt und überprüft wird, um den Besitz einer Domäne zu verifizieren.

Wie bereits bei der "normalen" HTTPS Konfiguration wird eine *.env* Datei angelegt, um die Optionen zu verwalten. Zuerst wird die Docker Compose Konfiguration wie folgt erstellt.

```
services:
  traefik:
    image: traefik
    container_name: traefik
    restart: unless-stopped
    ports:
      - 80:80
      - 443:443
      - 8080:8080
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ${TRAEFIK_PFAD}/traefik_data:/traefik
    command:
      - --api.dashboard=true
      - --api.insecure=true
      - --providers.docker=true
      - --providers.docker.exposedbydefault=false
      - --providers.docker.network=traefik
      - --entrypoints.webinsecure.address=:80
      - --entrypoints.websecure.address=:443
```

```
      - --entrypoints.traefik.address=:8080
      - --metrics.prometheus.addrouterslabels=true
      - --certificatesresolvers.wildcard-godaddy.acme.dnschallenge=true
      - --certificatesResolvers.wildcard-godaddy.acme.dnsChallenge.provider=godaddy
      - --certificatesResolvers.wildcard-godaddy.acme.dnsChallenge.delayBeforeCheck=0
      - --certificatesresolvers.wildcard-godaddy.acme.email=${ACME_EMAIL}
      - --certificatesresolvers.wildcard-godaddy.acme.storage=/traefik/certs/acme.json
      - --certificatesResolvers.wildcard-
godaddy.acme.dnsChallenge.resolvers=1.1.1.1:53,8.8.8.8:53
    labels:
      - traefik.enable=true
      # HTTPS Umleitung
      - traefik.http.routers.http-catchall.entrypoints=webinsecure
      - traefik.http.routers.http-catchall.rule=HostRegexp(`{host:.+}`)
      - traefik.http.routers.http-catchall.middlewares=redirect-to-https
      - traefik.http.middlewares.redirect-to-https.redirectscheme.scheme=https
      # HTTP Router
      - traefik.http.routers.traefik-secure.entrypoints=websecure
      - traefik.http.routers.traefik-secure.rule=Host(`traefik.${DOMAIN}`)
      - traefik.http.routers.traefik-secure.tls=true
      - traefik.http.routers.traefik-secure.tls.certresolver=wildcard-godaddy
      - traefik.http.routers.traefik-secure.tls.domains[0].main=*.${DOMAIN}
      - traefik.http.routers.traefik-secure.tls.domains[0].sans=${DOMAIN}
      - traefik.http.routers.traefik-secure.service=api@internal
    networks:
      - traefik

networks:
  traefik:
    name: traefik
    external: true
```

Diese kann im Prinzip genau so übernommen werden, wer mag, kann noch die Namen anpassen.

> In der oben gezeigten Konfiguration ist noch das Dashboard verfügbar, welches alle Interna von Traefik anzeigt. Für ein produktives Setup, dass von außen erreichbar ist, sollte dieses besser abgeschaltet werden. Dazu die folgenden Zeilen löschen:
>
> - 8080:8080
> - --entrypoints.traefik.address=:8080
> - traefik.http.routers.traefik-secure.rule=Host(`traefik.${DOMAIN}`)

Wichtig sind die Optionen, die in der folgenden *.env* Datei eingestellt werden. In dieser bitte alle Variablen anpassen.

```
GODADDY_API_KEY=abcdefghiklmnopqrstuvwxyz
GODADDY_API_SECRET=abcdefghijklmnopqrstuvw
LEGO_DISABLE_CNAME_SUPPORT=true
TRAEFIK_PFAD=/pfad/zu/traefik
ACME_EMAIL=mail@beispiel.de
DOMAIN=beispiel.de
```

Bevor nun die Konfiguration angewendet wird, müssen schon der Ordner traefik_data und darin der Unterordner für die Zertifikate erstellt werden. Zusätzlich ist die Datei für die Zertifikatskonfiguration zu erstellen und die Berechtigung anzupassen. Dazu kann das folgende Skript verwendet werden, wobei der Pfad anzupassen ist.

```
pfad=/pfad/zu/traefik/traefik_data/certs_einzelzertifikat
mkdir $pfad
touch mkdir $pfad/acme.json
chmod 600 $pfad/acme.json
```

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Anschließend müssen nur noch die gewünschten Dienst konfiguriert werden, damit diese von Traefik verwaltet werden können.

Dienst konfigurieren

Um nun den Dienst eines anderen Containers anzubieten, wird mit den Docker Labeln gearbeitet. Diese liest Traefik automatisch laufend aus und stellt die Dienste bereit, so wie sie in den Labeln beschrieben werden. Es folgt ein Beispiel-Dienst von Traefik selbst: whoami. Dieser zeigt lediglich einige Serverinformationen an, jedoch eignet er sich hervorragend dazu, um die automatische Konfiguration zu demonstrieren.

Zunächst erstellen wir eine neue Docker Compose Konfiguration mit folgendem Inhalt.

```
services:
  whoami:
    image: "traefik/whoami"
    container_name: "simple-service"
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.whoami.rule=Host(`whoami.localhost`)"
      - "traefik.http.services.whoami.loadbalancer.server.port=80"
    networks:
      - traefik

networks:
  traefik:
    name: traefik
    external: true
```

> Sofern der Dienst für HTTPS anstatt HTTP konfiguriert wird, sind folgende Label zusätzlich hinzuzufügen:
> traefik.http.routers.whoami.tls=true
> traefik.http.routers.whoami.tls.certresolver=einzelzertifikat
> Der Certresolver ist auf den Namen des Resolvers zu ändern, wie dieser in der Traefik Konfiguration angelegt wurde.

Hier sehen wir nun 3 Label für die Traefik Konfiguration.

1. Enable: damit wird dieser Container für Traefik markiert und von diesem verwaltet
2. Rule: hier wird festgelegt, welche Regel zum Aufruf des Dienstes führt, in diesem Fall durch den Aufruf über die Domäne whoami.localhost

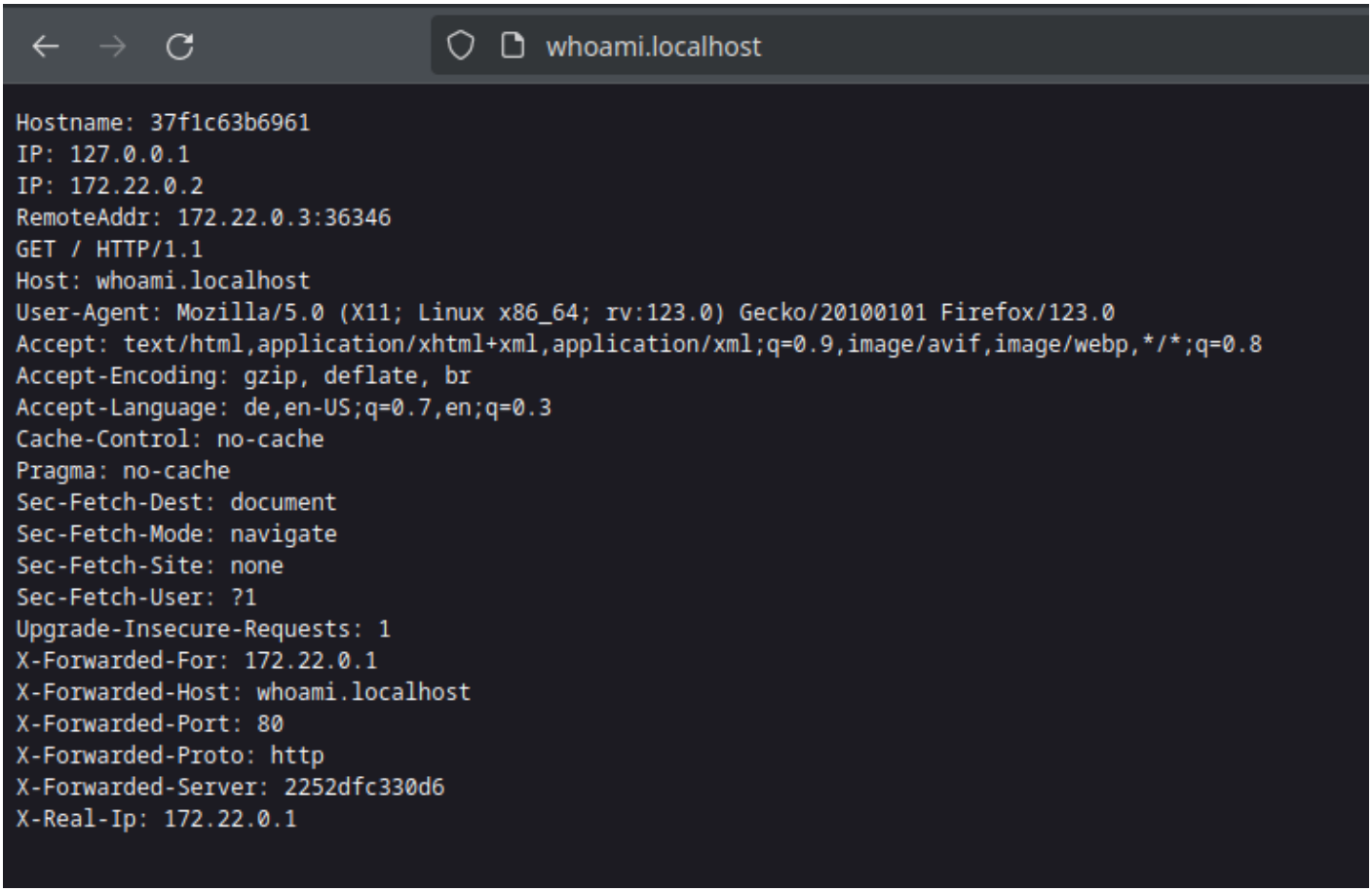
3. Port: Gibt den Port des Dienstes an, an welchen Traefik die Anfragen weiterleitet, sofern der Dienst nur einen Port bereitstellt, kann diese Option weggelassen werden.

> Bitte bei 2. auf die Anführungszeichen achten, es sind nicht die geraden einfachen, sondern die schrägen einzelnen Striche, die wie ein Backslash verlaufen.

Damit Traefik und Whoami miteinander kommunizieren können, wird Whoami in das Netzwerk aufgenommen, zu dem bereits Traefik hinzugefügt wurde. In diesem Beispiel heißt das Netzwerk *traefik*, wie bereits in der Installationsanleitung.

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Anschließend kann mit http://whoami.localhost der neue Dienst über den Reverse Proxy getestet werden.



> Besonders angenehm: Traefik muss nicht neu gestartet werden, sondern sobald der neue Container gestartet ist, wird dieser automatisch von Traefik erkannt und dieser konfiguriert sich selbst. Die Seite ist kurz danach bereits über Traefik erreichbar.

Dienst mit Basic Auth absichern

Traefik bietet ein Modul für Basic Auth an, um einzelne Dienste mit einem Zugangsschutz zu versehen.

Angenommen der Whoami Dienst, welcher unter Dienst konfigurieren erklärt wird, soll mit einem solchen Zugangsschutz versehen werden. Dann würde die Docker Compose Datei wie folgt erweitert werden.

```
services:
  whoami:
    image: "traefik/whoami"
    container_name: "simple-service"
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.whoami.rule=Host(`whoami.localhost`)"
      - "traefik.http.services.whoami.loadbalancer.server.port=80"
      - "traefik.http.middlewares.whoami-
auth.basicauth.users=benutzername:$2a$12$74NsF8MzGC25qO5tYoGfnO1Tg9I9c5Fcbu/zwi1/paDfdYD8eeUxy
"
      - "traefik.http.routers.whoami.middlewares=whoami-auth"
    networks:
      - traefik

networks:
  traefik:
    name: traefik
    external: true
```

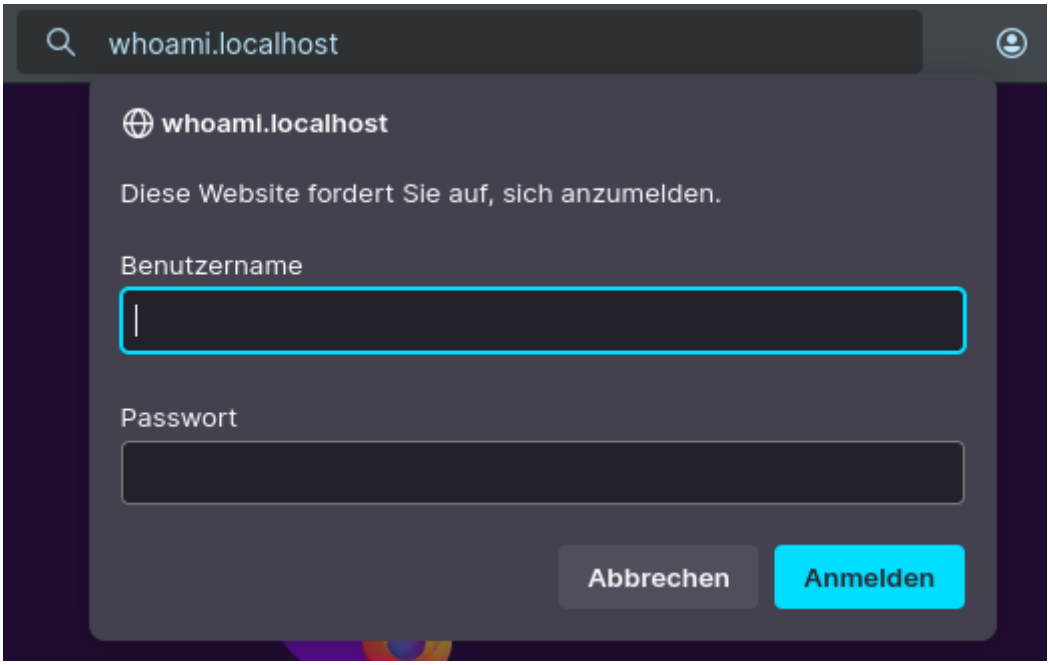
Die 1. neue Zeile gibt an, wie die Logindaten für den Benutzer bzw. die Benutzer sind. Außerdem wird ein Name für das Modul vergeben, in diesem Fall *whoami-auth*. Dann wird ein Benutzername (hier: benutzername) vergeben und nach dem Doppelpunkt wird das Passwort als Hash angegeben. Am besten wird der bcrypt Hash verwendet, welcher z. B. auf der folgenden Webseite generiert werden kann: Bcrypt-Generator

```
      - "traefik.http.middlewares.whoami-
auth.basicauth.users=benutzername:$2a$12$74NsF8MzGC25qO5tYoGfnO1Tg9I9c5Fcbu/zwi1/paDfdYD8eeUxy
"
```

Mit der 2. neuen Zeile wird das Modul auf den *whoami*-Dienst angewendet.

```
      - "traefik.http.routers.whoami.middlewares=whoami-auth"
```

Anschließend mit docker compose up -d nochmal die Container neu erstellen lassen und fertig. Ab jetzt ploppt ein Anmeldefenster auf, wenn auf die Seite zugegriffen wird.



Nur wenn die richtigen Logindaten eingegeben werden, wird die Seite angezeigt, ansonsten erscheint eine Fehlerseite mit dem Fehlercode *401 Unauthorized*.

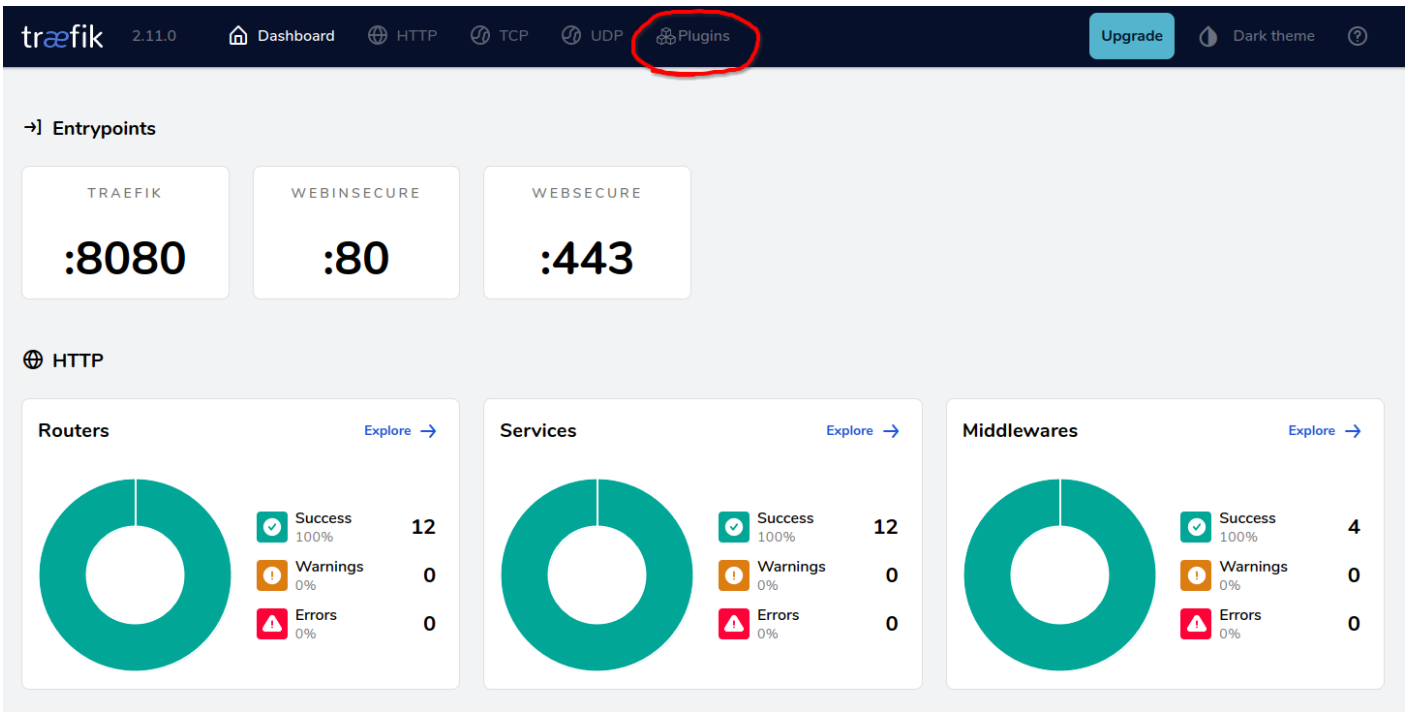
Plugins

Von der Community gibt es einige Plugins, die zusätzliche Funktionen zu Traefik hinzufügen. Ein paar interessante und hilfreiche Plugins werden hier vorgestellt, wie z. B. das Plugin GeoBlock von Pascal Minder.

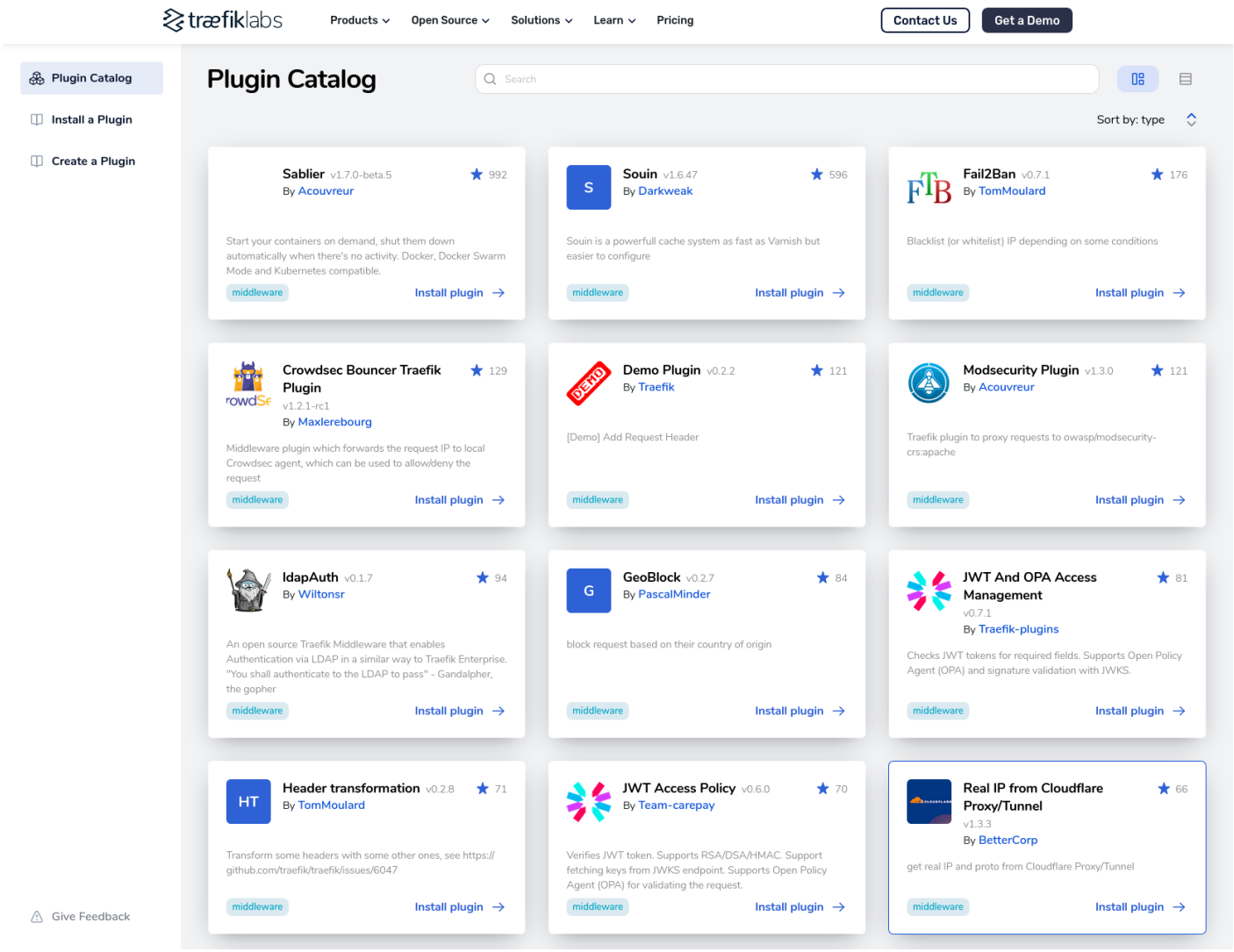
Plugin installieren

Die Installation eines Plugins ist wie folgt.

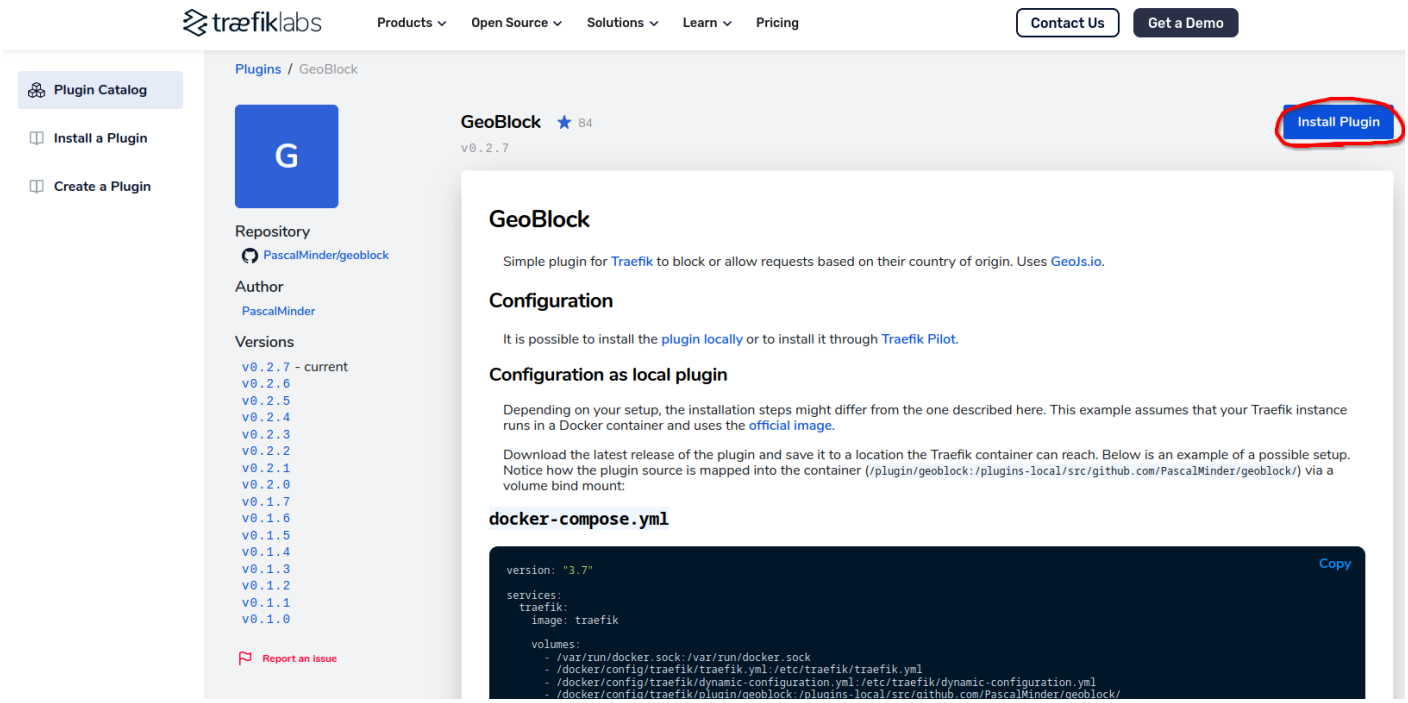
Für eine einfache Suche nach Plugins kann die offizielle Seite von Traefik genutzt werden, diese kann im Dashboard aufgerufen werden. Hierzu oben im Menüband auf *Plugins* klicken.



Nun öffnet sich der Plugin Dialog in einem neuen Tab. Hier kann nun nach dem gewünschten Plugin gesucht werden.



Wenn ein Plugin gefunden wurde, dieses einfach anklicken, um die Dokumentation des Plugins zu öffnen. In der Dokumentation wird auch eine *Install Plugin* Schaltfläche angezeigt, die eine kurze Installationsanweisung anzeigt.



Die angezeigten Schritte werden durchgeführt bzw. nur einige davon, wenn mehrere Installations und Konfigurationsmöglichkeiten angezeigt werden.

> Einige Beispiele, wie Plugins installiert und konfiguriert werden, finden sich auf den anderen Seiten in diesem Kapitel

How to install "GeoBlock" plugin?

1 Add this snippet in the Traefik Static Configuration:

File (YAML) File (TOML) CLI

```
--experimental.plugins.geoblock.modulename=github.com/PascalMinder/geoblock
--experimental.plugins.geoblock.version=v0.2.7
```

Copy

2 Configure the plugin using the Dynamic Configuration.
Example:

Kubernetes CRD File (YAML) File (TOML)

```yaml
http:
    middlewares:
        my-geoblock:
            plugin:
                geoblock:
                    allowLocalRequests: "false"
                    allowUnknownCountries: "false"
                    api: https://get.geojs.io/v1/ip/country/{ip}
                    apiTimeoutMs: "150"
                    cacheSize: "15"
                    countries:
                        - CH
                    forceMonthlyUpdate: "true"
                    logAllowedRequests: "false"
                    logApiRequests: "true"
                    logLocalRequests: "false"
                    silentStartUp: "false"
                    unknownCountryApiResponse: nil
```

Copy

Restart you Traefik instance and enjoy! 👌

GeoBlock von Pascal Minder

Ein super nützliches Plugin für alle, die häufig von Leuten aus anderen Ländern (z. B. Russland) belästigt werden, aber von dort keine Zugriffe erwarten: GeoBlock von Pascal Minder

Nehmen wir die Compose Konfiguration von der Seite Installation und den Testdienst von der Seite Dienst konfigurieren.

Die beiden Konfigurationen werden testweise zu einer Konfigurationsdatei zusammengeführt und dann um die Einstellungen für das Plugin erweitert. Somit sieht die neue Konfigurationsdatei nun wie folgt aus.

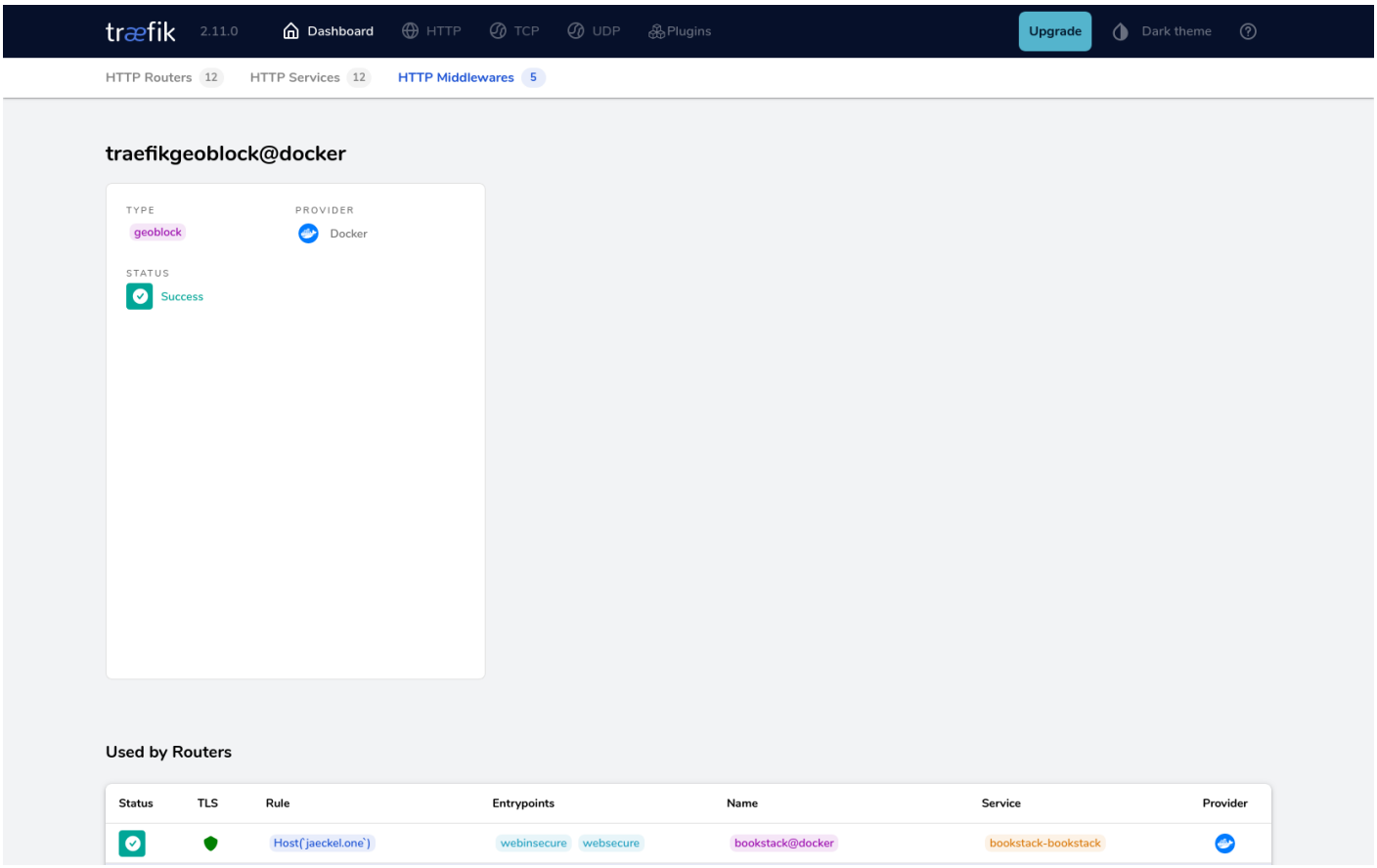
```
services:
  traefik:
    image: "traefik"
    container_name: "traefik"
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
# Plguin GeoBlock von Pascal Minder hinzufügen
      - --experimental.plugins.geoblock.modulename=github.com/PascalMinder/geoblock
      - --experimental.plugins.geoblock.version=v0.2.7
    labels:
#      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.allowLocalRequests=true
#      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.allowUnknownCountries=false
      -
traefik.http.middlewares.traefikgeoblock.plugin.geoblock.api=https://get.geojs.io/v1/ip/countr
y/{ip}
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.cacheSize=15
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.blackListMode=true
# Anfragen von russichen und chinesischen IPs blockieren
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.countries=RU,CN
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.forceMonthlyUpdate=true
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.logAllowedRequests=false
```

```
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.logApiRequests=false
      - traefik.http.middlewares.traefikgeoblock.plugin.geoblock.logLocalRequests=false
# Plugin Konfiguration Ende
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
    networks:
      - traefik

services:
  whoami:
    image: "traefik/whoami"
    container_name: "simple-service"
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.whoami.rule=Host(`whoami.localhost`)"
      - "traefik.http.services.whoami.loadbalancer.server.port=80"
# Plugin GeoBlock für whoami aktivieren
      - traefik.http.routers.whoami.middlewares=traefikgeoblock
# Plugin Aktivierung Ende
    networks:
      - traefik

networks:
  traefik:
    name: traefik
    external: true
```

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Nachdem die Konfiguration gestartet wurde, kann über das Traefik Dashboard auch das geladene Modul eingesehen werden.

Hierzu einfach im Dashboard auf Middlewares - Explore klicken und dann traefikgeoblock@docker auswählen. Die Ansicht sieht dann in etwa wie folgt aus. Ganz unten werden alle Router dargestellt, die dieses Plugin verwenden.



Dynamische Konfiguration

Traefik bietet verschiedene Möglichkeiten der Konfiguration. Bisher wurden alle Konfigurationen über die Docker Compose Datei vorgenommen. Stattdessen kann auch eine dynamische Konfiguration in Form einer oder mehrerer YAML-Datei(en) verwendet werden.

Der Große Vorteil dieser YAML-Dateien liegt in der Möglichkeit, Traefik ohne Neustart anzupassen. Einfach die Datei(en) ändern und Traefik erkennt automatisch die Änderungen und lädt seine Konfiguration neu. Hierdurch gibt es keine Verbindungsabbrüche.

Zuerst erstellen wir einen Ordner samt Unterordner im Verzeichnis der Docker Compose Datei, z. B. mit den Namen *traefik_data/conf* und wechseln in dieses.

```
mkdir traefik_data
mkdir traefik_data/conf
cd traefik_data/conf
```

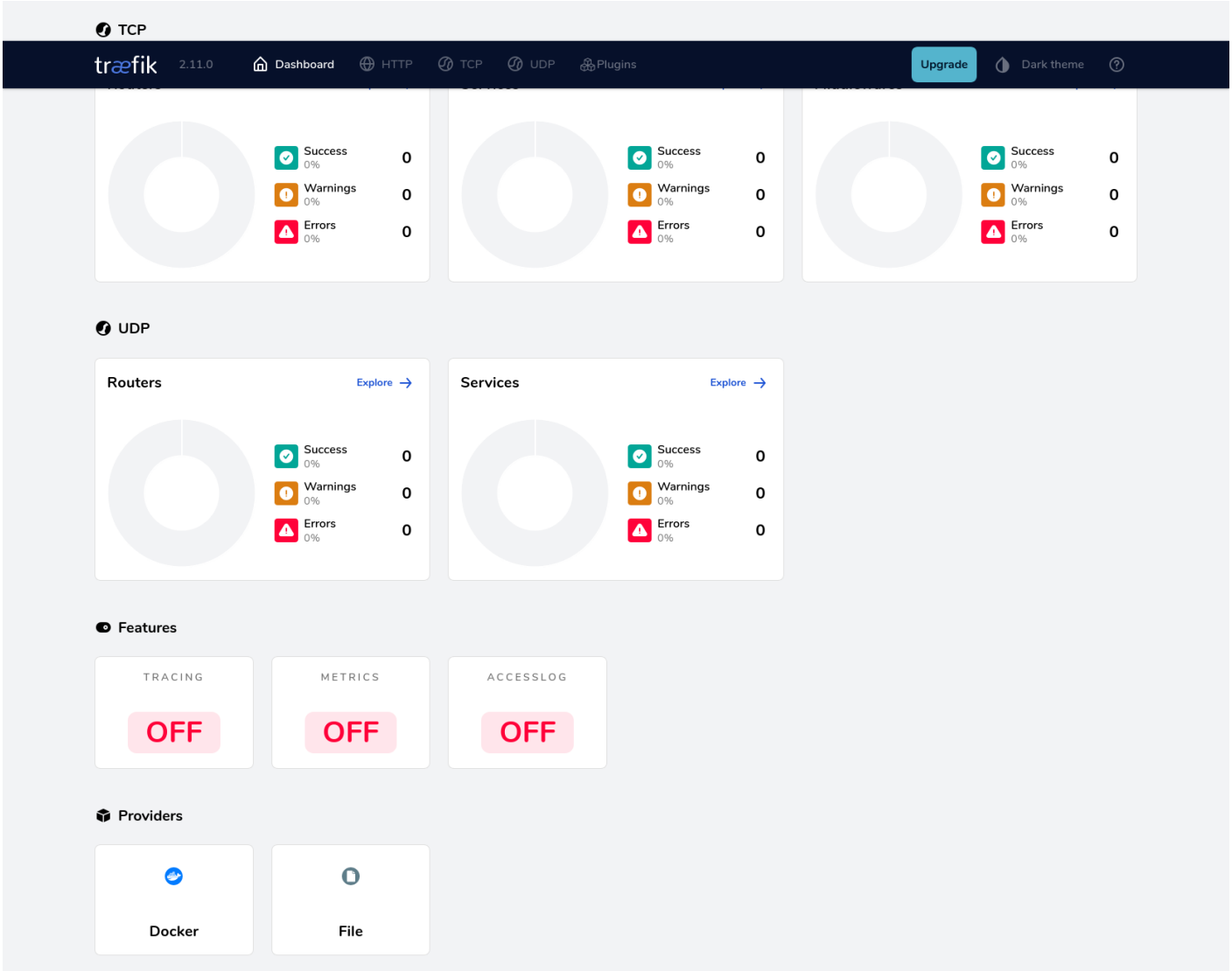
Als Grundlage nehmen wir die Compose Konfiguration von der Seite Installation und erweitern diese um ein Volume, in welchem wir später die dynamischen Konfigurationsdateien ablegen. Zusätzlich ist unter Command noch der Pfad zu dem Ordner mit den Konfigurationsdateien im Container anzugeben. Der soeben erstellte Ordner wird als Volume eingebunden.

```
services:
  traefik:
    image: "traefik"
    container_name: "traefik"
    command:
      - "--providers.file.directory=/traefik/conf"
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--providers.docker.network=traefik"
      - "--entrypoints.web.address=:80"
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
```

```
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /pfad/zu/traefik/traefik_data:/traefik
    networks:
      - traefik
networks:
  traefik:
    external: true
```

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Nachdem Traefik läuft, einmal die korrekte Ausführung prüfen, entweder durch den Aufruf von http://localhost:8080 (Dashboard) oder indem in die Docker Logs geschaut wird. Im Dashboard kann unten auf der Seite unter *Providers* geprüft werden, ob der Ordner für die Konfigurationsdateien überwacht wird. Dort sollte neben Docker nun auch File stehen.



Nun fügen wir einen Dienst über eine Konfigurationsdatei hinzu. Am besten nehmen wir hierzu das Beispiel von der Seite Dienst mit Basic Auth absichern.

Angenommen wir wollen mehrere Dienste mit Basic-Auth absichern, dann wäre es am besten, die Konfiguration der Benutzer in einer dynamischen Konfigurationsdatei vorzunehmen. Wir streichen also die Zeile traefik.http.middlewares.whoami-auth.basicauth.users aus der Compose Datei und ändern den Namen der Zeile für *middlewares* ab. Die neue Compose Datei sieht wie folgt aus.

```
services:
  whoami:
    image: "traefik/whoami"
    container_name: "simple-service"
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.whoami.rule=Host(`whoami.localhost`)"
      - "traefik.http.services.whoami.loadbalancer.server.port=80"
      - "traefik.http.routers.whoami.middlewares=basic-auth@file"
```

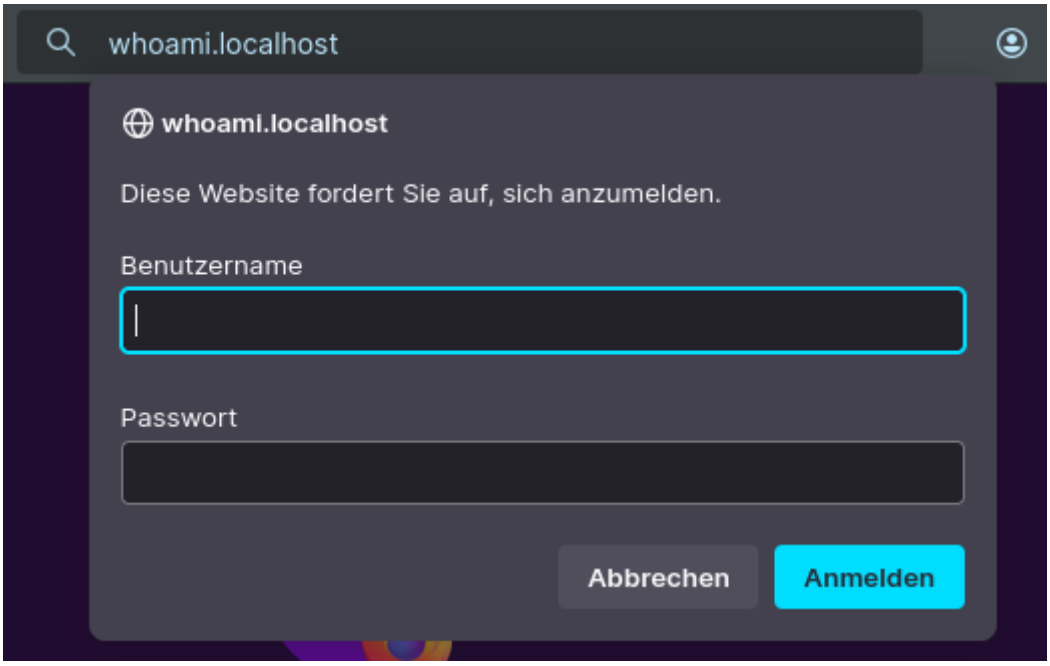
```
    networks:
      - traefik
networks:
  traefik:
    name: traefik
    external: true
```

Die Compose Konfiguration kann nun gestartet werden.

Anschließend erstellen wir die Datei `traefik_data/conf/basic-auth.yml` mit folgendem Inhalt.

```
http:
  middlewares:
    basic-auth:
      basicauth:
        users:
          benutzername:$2a$12$74NsF8MzGC25qO5tYoGfnO1Tg9I9c5Fcbu/zwi1/paDfdYD8eeUxy
```

Nun können wir direkt den neuen Dienst http://whoami.localhost aufrufen und es erscheint eine Login Abfrage.



Für die Generierung des Passworts kann folgende Seite verwendet werden: Bcrypt-Generator

Es können beliebige weitere Dateien erstellt werden um Plugins oder Seiten oder Dienst usw. zu konfigurieren. Die Einstellungen werden direkt angewendet, sobald die Datei gespeichert wird. Sollten Syntax-Fehler gefunden werden, bleibt die alte Konfiguration bestehen und im Docker Log werden Fehler angezeigt.