

Docker

Beschreibung von Docker Anwendungen

- [mealie Online-Kochbuch](#)
- [docker-compose](#)
- [Glances ein Raspberry Pi System Monitor](#)
 - [Installation auf einem Raspberry Pi 5](#)
 - [Installation auf einem Raspberry Pi 4 mit ARMV7 Prozessor](#)
- [openwebrx plus](#)
- [Nvidia Toolkit installieren](#)
- [docker expose](#)
- [wireguard](#)

mealie Online-Kochbuch

Es gibt ein sehr interessante Online Browser basierendes Kochbuch "Mealie"

🔍 Ergebnisse

🎲 Zufall ⋮



schnelles Gnocchi Rezept aus der P...

♥ ☆ ☆ ☆ ☆ ☆



Grillhähnchen

♥ ☆ ☆ ☆ ☆ ☆



Curry-Hähnchenbrust mit Quinoa un...

♥ ☆ ☆ ☆ ☆ ☆



Steckrüben-Spaghetti

♥ ☆ ☆ ☆ ☆ ☆



Kirschmichel

♥ ☆ ☆ ☆ ☆ ☆

Installation mit Docker-Compose

Docker-Compose - hier klicken

```
version: "3.7"
services:
  mealie-frontend:
    image: hkotel/mealie:frontend-nightly
    container_name: mealie-frontend
    depends_on:
      - mealie-api
    environment:
      # Set Frontend ENV Variables Here
      - API_URL=http://mealie-api:9000 #
    restart: always
```

ports:

- "9926:3000" #

volumes:

- ./data:/app/data/ #

mealie-api:

image: hkotel/mealie:api-nightly

container_name: mealie-api

depends_on:

- postgres

volumes:

- ./data:/app/data/

environment:

Set Backend ENV Variables Here

- ALLOW_SIGNUP=true
- PUID=1000
- PGID=1000
- TZ=Europe/Berlin
- MAX_WORKERS=1
- WEB_CONCURRENCY=1
- BASE_URL=https://kochbuch.xxxx.de

Database Settings

- DB_ENGINE=postgres
- POSTGRES_USER=mealie
- POSTGRES_PASSWORD=mealie
- POSTGRES_SERVER=postgres
- POSTGRES_PORT=5432
- POSTGRES_DB=mealie

restart: always

postgres:

container_name: postgres

image: postgres:15

restart: always

environment:

POSTGRES_PASSWORD: mealie

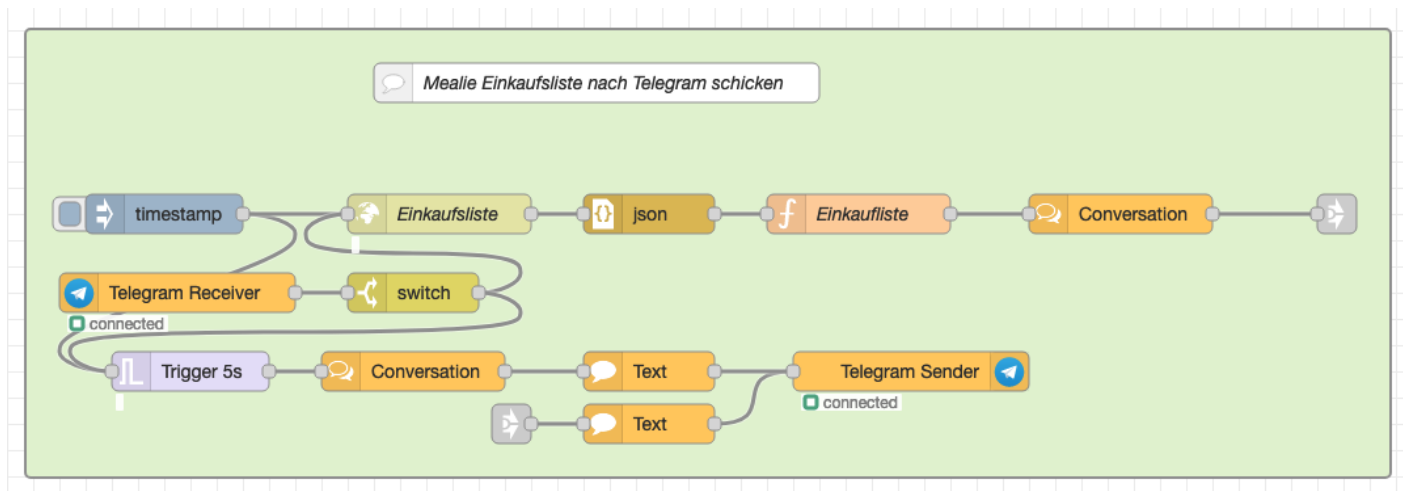
POSTGRES_USER: mealie

Einkaufsliste nach Telegram schicken

Benötigt wird folgendes Plugin:

<https://flows.nodered.org/node/node-red-contrib-chatbot> braucht NPM 8.19.4 und Node v16.20.0

HINWEIS: Die Palette kann nicht in Docker Container <https://hub.docker.com/r/nodered/node-red/> installiert da die oben genannten Version im Image nicht vorhanden sind. Man muss also selbst ein eigenes Images bauen oder NodeRED native auf einem Linux Container installiert werden.



NodeRED Code

NodeRED Code - hier klicken

```
[
  {
    "id": "767e8ab9a03f0e4f",
    "type": "inject",
    "z": "abcd6ab2be488b84",
    "name": "",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",
        "vt": "str"
      }
    ]
  },
  {
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
```

```
"topic": "",
"payload": "",
"payloadType": "date",
"x": 200,
"y": 240,
"wires": [
  [
    "626bb8984931c788",
    "94531acc91568a22"
  ]
],
},
{
  "id": "626bb8984931c788",
  "type": "trigger",
  "z": "abcd6ab2be488b84",
  "name": "",
  "op1": "Einkaufliste",
  "op2": "0",
  "op1type": "str",
  "op2type": "str",
  "duration": "5",
  "extend": true,
  "overrideDelay": false,
  "units": "s",
  "reset": "",
  "bytopic": "all",
  "topic": "topic",
  "outputs": 1,
  "x": 220,
  "y": 360,
  "wires": [
    [
      "fdab0ec18a7364d8"
    ]
  ]
},
{
  "id": "94531acc91568a22",
```

```
"type": "http request",
"z": "abcd6ab2be488b84",
"name": "Einkaufsliste",
"method": "GET",
"ret": "txt",
"paytoqs": "ignore",
"url": "https://kochbuch.unixweb.eth64.de/api/groups/shopping/items?group_id=XXXX-XXXXX-XXXXX-XXXXX&page=1&perPage=50&orderBy=created_at&orderDirection=desc&paginationSeed=Einkaufen",
"tls": "",
"persist": false,
"proxy": "",
"insecureHTTPParser": false,
"authType": "bearer",
"senderr": false,
"headers": [],
"x": 410,
"y": 240,
"wires": [
  [
    "b73ba28428b11da8"
  ]
]
},
{
  "id": "e7810838c682396c",
  "type": "switch",
  "z": "abcd6ab2be488b84",
  "name": "",
  "property": "payload.content",
  "propertyType": "msg",
  "rules": [
    {
      "t": "eq",
      "v": "/liste",
      "vt": "str"
    }
  ],
  "checkall": "true",
  "repair": false,
```

```
"outputs": 1,
"x": 390,
"y": 300,
"wires": [
  [
    "94531acc91568a22",
    "626bb8984931c788"
  ]
]
},
{
  "id": "fdab0ec18a7364d8",
  "type": "chatbot-conversation",
  "z": "abcd6ab2be488b84",
  "name": "",
  "botDevelopment": "83fad7bd2c994193",
  "botProduction": "83fad7bd2c994193",
  "chatId": "617681859",
  "userId": "",
  "transport": "telegram",
  "x": 390,
  "y": 360,
  "wires": [
    [
      "85f263c2ecc9df03"
    ]
  ]
},
{
  "id": "b73ba28428b11da8",
  "type": "json",
  "z": "abcd6ab2be488b84",
  "name": "",
  "property": "payload",
  "action": "",
  "pretty": true,
  "x": 570,
  "y": 240,
  "wires": [
```

```
[
    "fc2f6cc52c469909"
]
],
{
    "id": "c5b7376177890472",
    "type": "chatbot-telegram-receive",
    "z": "abcd6ab2be488b84",
    "bot": "",
    "botProduction": "",
    "x": 210,
    "y": 300,
    "wires": [
        [
            "e7810838c682396c"
        ]
    ]
},
{
    "id": "85f263c2ecc9df03",
    "type": "chatbot-message",
    "z": "abcd6ab2be488b84",
    "name": "",
    "message": [
        {
            "message": "Einkaufsliste -->>>>>>>>>>>>>>>"
        }
    ],
    "language": "none",
    "x": 570,
    "y": 360,
    "wires": [
        [
            "90658281a5e95487"
        ]
    ]
},
{
```



```
"id": "fc2f6cc52c469909",
"type": "function",
"z": "abcd6ab2be488b84",
"name": "Einkaufliste",
"func": "for (var i = 0; i < msg.payload.total; i++) {\n  var newMsg = {};\n  newMsg.payload =
msg.payload.items[i].note;\n  node.send(newMsg);\n}\nreturn null;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 730,
"y": 240,
"wires": [
  [
    "81eaa91b4f669350"
  ]
]
},
{
  "id": "90658281a5e95487",
  "type": "chatbot-telegram-send",
  "z": "abcd6ab2be488b84",
  "bot": "",
  "botProduction": "",
  "track": false,
  "passThrough": false,
  "errorOutput": false,
  "outputs": 0,
  "x": 770,
  "y": 360,
  "wires": []
},
{
  "id": "81eaa91b4f669350",
  "type": "chatbot-conversation",
  "z": "abcd6ab2be488b84",
  "name": "",
  "botDevelopment": "83fad7bd2c994193",
```

```
"botProduction": "83fad7bd2c994193",
"chatId": "XXXXXXXXXX",
"userId": "",
"transport": "telegram",
"x": 930,
"y": 240,
"wires": [
  [
    "4d675f4164517f5d"
  ]
],
},
{
  "id": "5b4f462150cf7432",
  "type": "chatbot-message",
  "z": "abcd6ab2be488b84",
  "name": "",
  "message": [
    {
      "message": ""
    }
  ],
  "language": "",
  "x": 570,
  "y": 400,
  "wires": [
    [
      "90658281a5e95487"
    ]
  ]
},
{
  "id": "4d675f4164517f5d",
  "type": "link out",
  "z": "abcd6ab2be488b84",
  "name": "link out 4",
  "mode": "link",
  "links": [
    "07d30318e27ee772"
```

```
    ],
    "x": 1095,
    "y": 240,
    "wires": []
  },
  {
    "id": "07d30318e27ee772",
    "type": "link in",
    "z": "abcd6ab2be488b84",
    "name": "link in 1",
    "links": [
      "4d675f4164517f5d"
    ],
    "x": 465,
    "y": 400,
    "wires": [
      [
        "5b4f462150cf7432"
      ]
    ]
  }
]
```

Die Gruppen-ID ist in der Gruppenverwaltung von Mealie zu finden.

Für den **API-Aufruf** gibt man die URL mit **/docs** am Ende ein. Dort ist eine Restful API vorhanden, die Parameter selbst zusammen bauen kann. Dort können weitere Funktionen implementiert werden mit NodeRED.

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11

15:11



docker-compose

Die Tage hab ich mich gewundert, als der docker-compose beim Deployment massive Fehler ausgegeben hatte.

Er konnte keine Dateien anlegen, obwohl die Rechte alle gepasst hatten (gibt auch noch einen anderen Grund aber nicht in diesem Fall). Nach etwas Suche ergab sich dass das in den jeweiligen Linux Distributionen enthaltene docker-compose total veraltet ist und es keine aktuellen Packages für die jeweilige Distribution gibt.

Inzwischen gibt es ein neues Software Paket "docker-compose-v2" welches mit "apt install docker-compose-v2" installiert werden kann. Es muss "docker-compose" deinstalliert werden, damit der Befehl fehlerfrei auf dem Dockerhost ausgeführt werden kann.

Deshalb gibt es einen anderen Weg.

Aktuelle Umgebung :

```
docker-compose -v
docker-compose version 1.29.2, build unknown
joachim@dockermachine:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
```

Wenn man die Seite besucht: <https://github.com/docker/compose> gibt es bereits ein Release 2.24.1

Neues Verfahren :

Das Docker Compose wird als CLI-Plugin in Docker eingebunden. Dh. in Zukunft gibt man nicht mehr "docker-compose -f docker-compose.yml up -d" ein, sondern "**docker compose -f docker-compose.yml up -d**".

```
DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
mkdir -p $DOCKER_CONFIG/cli-plugins
curl -SL https://github.com/docker/compose/releases/download/v2.24.0/docker-compose-linux-x86_64 -o
$DOCKER_CONFIG/cli-plugins/docker-compose
```

```
chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
```

Die Installation erfolgt im \$HOME des angemeldeten Benutzer der "NICHT ROOT" ist. Der Benutzer muss in der Gruppe "docker" sein.

```
usermod -aG docker <USERNAME>
```

Sollte Docker Compose für alle Benutzer auf dem Dockerhost zur Verfügung sein muss es in folgenden Verzeichnis installiert werden :

```
sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
```

Nun folgenden Befehl eingeben:

```
$ docker compose version  
Docker Compose version v2.24.0
```

Ab jetzt sollten Docker Anwendungen mit docker-compose.yml Files nur noch mit folgenden Befehl deployed werden:

```
docker compose -f docker-compose.yml up -d
```

Und wie ein Wunder, es sind alle Fehlermeldungen verschwunden.

Es gibt Zusatztools wie : `apt install docker-ce docker-ce-cli docker-compose-plugin`

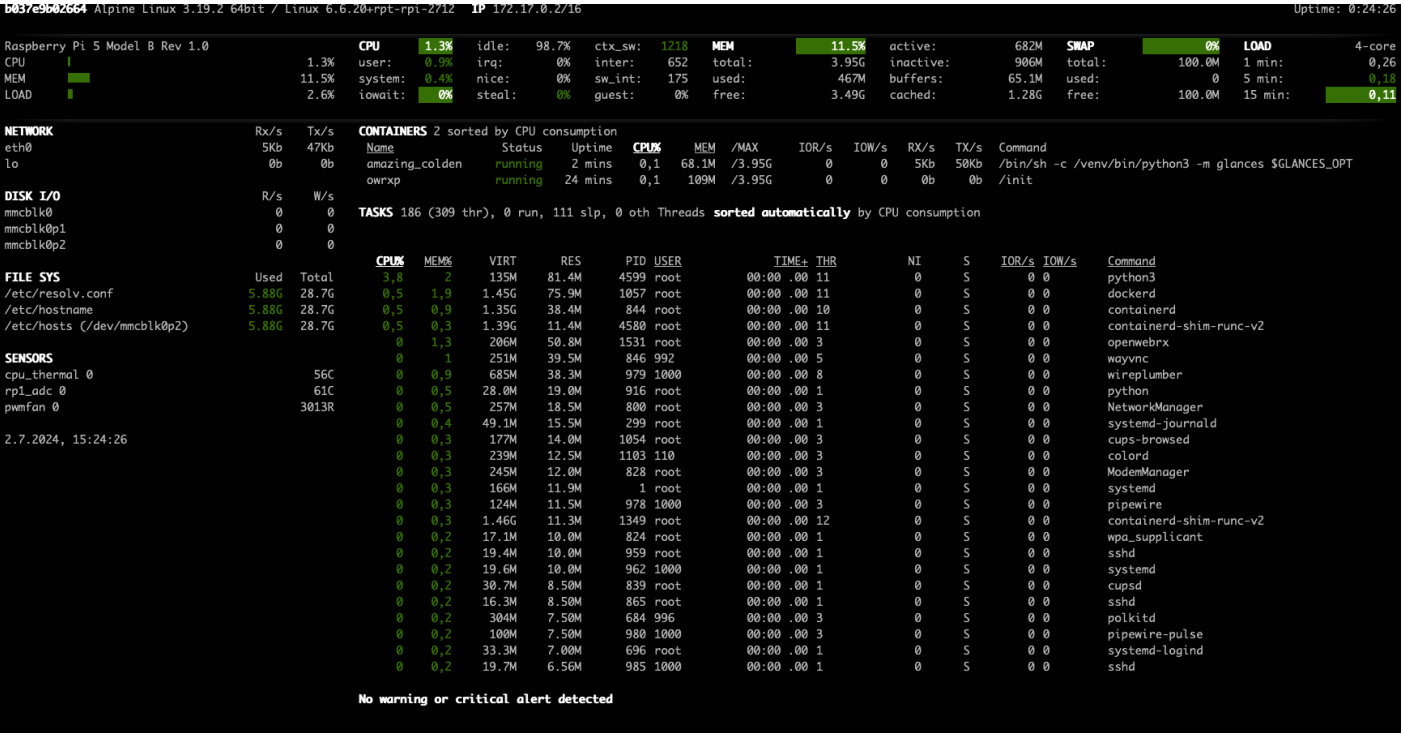
Diese funktionieren aber nicht oder sind zumindest bei Ubuntu 20.04 / 22.04 nicht mehr vorhanden, deshalb muss dieser Weg so gemacht werden. Ich vermute mal dass docker.io so ein ähnliches Problem hat, mal sehen ich werde da mal auf Suche gehen.

Glances ein Raspberry Pi System Monitor

Glances mit Docker installieren

Installation auf einem Raspberry Pi 5

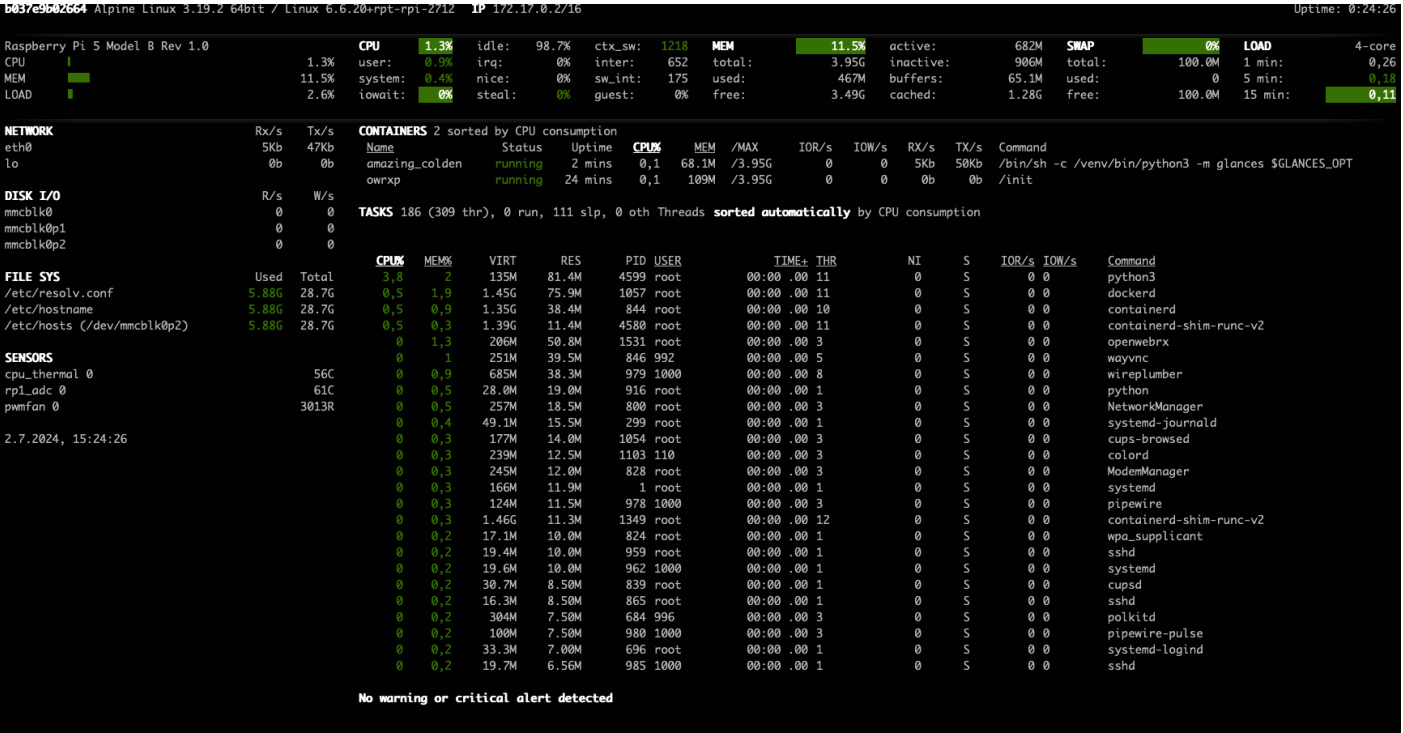
Glances ist ein plattformübergreifendes Systemüberwachungstool, das eine umfassende Echtzeit-Übersicht über verschiedene Systemmetriken wie CPU-, Speicher- und Netzwerkauslastung bietet. Es ist flexibel und erweiterbar, unterstützt mehrere Ausgabeformate und kann sowohl im Terminal als auch über ein Web-Interface genutzt werden.



```
docker run -d --restart="always" -p 61208-61209:61208-61209 -e GLANCES_OPT="-w" -v /var/run/docker.sock:/var/run/docker.sock:ro --pid host nicolargo/glances:latest-full
```

Installation auf einem Raspberry Pi 4 mit ARMV7 Prozessor

Glances ist ein plattformübergreifendes Systemüberwachungstool, das eine umfassende Echtzeit-Übersicht über verschiedene Systemmetriken wie CPU-, Speicher- und Netzwerkauslastung bietet. Es ist flexibel und erweiterbar, unterstützt mehrere Ausgabeformate und kann sowohl im Terminal als auch über ein Web-Interface genutzt werden.



```
services:
  monitoring:
    image: nicolargo/glances:3.4.0.5
    restart: always
    pid: host
    ports:
```

- 61208-61209:61208-61209

volumes:

- /var/run/docker.sock:/var/run/docker.sock
- /etc/os-release:/etc/os-release:ro

environment:

- "GLANCES_OPT=-w"

Hier ist darauf zu achten dass die Version 3.4.0.5 genutzt wird. Die neueren Docker Container Versionen laufen nicht auf einem ARMV7 Prozessor.

openwebrx plus

openwebrx plus

services:

openwebrxplus-softmbe:

container_name: owrxp

devices:

- /dev/bus/usb

ports:

- 8073:8073

volumes:

- owrxp-settings:/var/lib/openwebrx

- owrxp-etc:/etc/openwebrx

privileged: true

restart: unless-stopped

image: slechev/openwebrxplus-softmbe

volumes:

owrxp-settings:

external: true

name: owrxp-settings

owrxp-etc:

external: true

name: owrxp-etc

Nvidia Toolkit installieren

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey -o /tmp/nvidia-gpgkey
```

Dearmor the GPG key and save it

```
# gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg /tmp/nvidia-gpgkey
```

Download the NVIDIA container toolkit list file

```
$ curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list -o /tmp/nvidia-list
```

Modify the list file to include the signature

```
# sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' /tmp/n
```

Update the package database

```
# apt-get update
```

After executing these commands, you've set the stage for the NVIDIA Container Toolkit, which will be vital in our next steps to fully integrate the CUDA Toolkit within a Docker container. #

```
ubuntu-drivers devices  
ubuntu-drivers list
```

docker expose

Docker Container ohne Ports betreiben.

Netzwerk einrichten damit die Zuweisung mit IP-Adressen funktioniert:

```
docker network create \
  --driver=bridge \
  --subnet=172.26.0.0/24 \
  my_bridge_network
```

Einrichtung Portainer:

```
services:
  portainer-ce:
    container_name: portainer
    restart: always
    volumes:
      - '/var/run/docker.sock:/var/run/docker.sock'
      - 'portainer_data:/data'
    image: 'portainer/portainer-ce:latest'
    networks:
      my_bridge_network:
        ipv4_address: 172.26.0.4

volumes:
  portainer_data:

networks:
  my_bridge_network:
    external: true
```

Einrichtung Apache2:

```
services:
  apache:
    image: httpd:latest
    container_name: apache2
```

```
volumes:
  - ./app:/usr/local/apache2/htdocs
```

```
networks:
  my_bridge_network:
    ipv4_address: 172.26.0.3
```

```
volumes:
  open-webui:
```

```
networks:
  my_bridge_network:
    external: true
```

Einrichtung Zoraxy:

```
services:
  zoraxy:
    image: zoraxydocker/zoraxy:latest
    container_name: zoraxy
    restart: unless-stopped
    ports:
      - 80:80
      - 443:443
    expose:
      - 8000:8000
    volumes:
      - /home/user/docker/zoraxy/config:/opt/zoraxy/config/
      - /home/user/docker/zerotier/config:/var/lib/zerotier-one/
      - /var/run/docker.sock:/var/run/docker.sock
      - /etc/localtime:/etc/localtime
    environment:
      FASTGEOIP: "true"
      ZEROTIER: "true"




    networks:
      my_bridge_network:
        ipv4_address: 172.26.0.2
```

```
networks:
```

```
my_bridge_network:
  external: true
```

HTTP Proxy

Proxy HTTP server with HTTP or HTTPS for multiple hosts. If you are only proxying for one host / domain, use Default Site instead.

Host	Destination	Virtual Directory
docker.businesshelpdesk.net 	172.26.0.4:9000	No Virtual Directory
web.businesshelpdesk.net 	172.26.0.2:8000	No Virtual Directory
www.businesshelpdesk.net 	172.26.0.3:80	No Virtual Directory

wireguard

```
tracert www.google.de
```

traceroute to www.google.de (216.58.206.35), 30 hops max, 60 byte packets

```
14 192.178.74.165 (192.178.74.165) 29.742 ms 29.720 ms 192.178.74.163 (192.178.74.163)
27.842 ms
```

```
15 1hr35s10-in-f3.1e100.net (216.58.206.35) 29.568 ms 29.519 ms 25.746 ms
```

```
pi@wireguard-test:~ $ netstat -rn
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
0.0.0.0	192.168.10.1	0.0.0.0	UG	0 0	0	eth0	
10.8.0.0	0.0.0.0	255.255.255.0	U	0 0	0	wg0	
192.168.10.0	0.0.0.0	255.255.255.0	U	0 0	0	eth0	

[Interface]

PrivateKey = Odfgdgdgdgdgsfgdgsfgertfg34g34g34Z6x14=

Address = 10.8.0.6/24

DNS = 192.168.10.2

MTU = 1420

[Peer]

PublicKey = idgfgdgdgdgdgdgdgdgdgdgdgdgdgdgc=

PresharedKey = 77Hh1XEDgdgfdgdgdsdgdgfdkfaGBci2Rk=

AllowedIPs = 10.8.0.0/24. # Wichtig damit 192.168.10.X erreichbar bleibt im lokalen Netz

PersistentKeepalive = 25

Endpoint = xxx.dnsfor.me:51820