Docker

Installation

install current version (ubutnu/debian)

```
curl -sSL https://get.docker.com | sh
sudo usermod -aG docker $USER
newgrp docker # to add docker in your current shell instance
```

execute docker without sudo

- `sudo groupadd docker` if not already exists
- `sudo usermod -aG docker $USER` add user to group
- logout and login, or restart on VM

proxy settings

create file in `/etc/systemd/system/docker.service.d/http-proxy.conf`

```
[Service]
Environment="HTTPS_PROXY=http://www-zproxy.myProxy.com:80/"
Environment="HTTP_PROXY=http://www-zproxy.myProxy.com:80/"
Environment="NO_PROXY=localhost,127.0.0.1,myProxy.com,10.0.0.0/8"
```

than execute

```
systemctl daemon-reload
systemctl restart docker


systemctl show --property Environment docker
```

Docker CLI

container

- `docker ps` shows all running container
  - `docker ps -a` shows all existing container
- `docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]`
  - `--name <container-name>` gives the created container a custom name
  - `--rm` removes the container after stoping
  - ~~`-d` detached mode - stops the container after main process exits~~ usually not desired
  - `-it` for interactive processes - keep STDIN open (`i`), allocate a pseudo-tty (`t`)
- typical example for testing: `docker run -it --rm --name test php:7.4 bash`
- use bash in running container: `docker exec -it containerName bash`

network

- list networks `docker network ls`
- add a network connection `docker network connect [OPTIONS] NETWORK CONTAINER`

logs

`docker logs --follow <container ID>`

- `--follow` streams logs until command is killed (`^c`)
- `--until=3s` streams logs for 3 seconds
- `--since 2019-03-02` show logs since specific date

build dockerfile

`docker build .`

- arguments
  - `.` build context (configs, mounts, ...)
  - `-f dockerfile` specifiy a custom dockerfile (independend from the build context)
  - `-t vendor/image-name:version` specify a name (and a version) for the build image
  - `--no-cache` build image without using caches
  - `--build-arg var_name=test` set ARG
- most used: `docker build -t draab/image-name:latest .`

best practice

- use **user** flag: `--user $(id -u):$(id -g)` to avoid problems on creating and accessing files

Projects

General

https://github.com/awesome-selfhosted/awesome-selfhosted

portainer

```
docker run -d -p 8000:8000 -p 9443:9443 -p 9000:9000 \
--name portainer \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v ./data/portainer:/data \
 portainer/portainer-ce:lts
```

```
services:
  portainer:
    image: portainer/portainer-ce:lts
    container_name: portainer
    restart: always
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./data/portainer:/data
    networks:
      - nginx

networks:
  nginx:
    external: true
```

Vaultwarden

A slim bitwarden project

```
services:
  portainer:
```

```
    image: vaultwarden/server:latest
    container_name: vaultwarden
    restart: always
#    ports:
#      - 80:80
    volumes:
      - ./data/vaultwarden:/data
    networks:
      - nginx

networks:
  nginx:
    external: true
```

Jupyter Notebook

https://jupyter-docker-stacks.readthedocs.io/en/latest/

`docker run -it --rm -p 8888:8888 -v "${PWD}":/home/jovyan/work jupyter/base-notebook`

Unify

[github docker unify]

```
services:
  unifi-network-application:
    image: lscr.io/linuxserver/unifi-network-application:latest
    container_name: unifi-network-application
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
      - MONGO_USER=unifi
      - MONGO_PASS=
      - MONGO_HOST=unifi-db
      - MONGO_PORT=27017
      - MONGO_DBNAME=unifi
      - MONGO_AUTHSOURCE=admin
      - MEM_LIMIT=1024 #optional
```

```
      - MEM_STARTUP=1024 #optional
      - MONGO_TLS= #optional
    volumes:
      - /path/to/unifi-network-application/data:/config
    ports:
      - 8443:8443
      - 3478:3478/udp
      - 10001:10001/udp
      - 8080:8080
      - 1900:1900/udp #optional
      - 8843:8843 #optional
      - 8880:8880 #optional
      - 6789:6789 #optional
      - 5514:5514/udp #optional
    restart: unless-stopped
  unifi-db:
    image: docker.io/mongo:<version tag>
    container_name: unifi-db
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=
      - MONGO_USER=unifi
      - MONGO_PASS=
      - MONGO_DBNAME=unifi
      - MONGO_AUTHSOURCE=admin
    volumes:
      - /path/to/data:/data/db
      - /path/to/init-mongo.sh:/docker-entrypoint-initdb.d/init-mongo.sh:ro
    restart: unless-stopped
```

Proxy

traefik

links

- youtube tutuorial
- traefik boilerplate (Lempa)

labels for connected containers

```
services:
  demo
    labels:
      - traefik.enable=true
      - traefik.http.services.demo.loadbalancer.server.port=9000
      - traefik.http.routers.demo.service=demo
      - traefik.http.routers.demo.entrypoints=websecure
      - traefik.http.routers.demo.rule=Host(`demo.draab.at`)
      - traefik.http.routers.demo.tls=true
      - traefik.http.routers.demo.tls.certresolver=cloudflare # -> netcup
    networks:
      - traefik


networks:
  traefik:
    external: true
```

nginx proxy manager

```
services:
  app:
    container_name: 'nginx-proxy-manager-app'
    image: 'jc21/nginx-proxy-manager:latest'
```

```
    restart: always
    ports:
      - '80:80'
#      - '81:81'
      - '443:443'
    environment:
      DB_MYSQL_HOST: "db"
      DB_MYSQL_PORT: 3306
      DB_MYSQL_USER: "npm"
      DB_MYSQL_PASSWORD: "npm!m0stSecretPa55w0rd"
      DB_MYSQL_NAME: "npm"
    volumes:
      - ./data/data:/data
      - ./data/letsencrypt:/etc/letsencrypt
    networks:
      - nginx
      - npm-internal
  db:
    container_name: 'nginx-proxy-manager-db'
    image: 'jc21/mariadb-aria:latest'
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 'npm!m0stSecret!r00t!Pa55w0rd'
      MYSQL_DATABASE: 'npm'
      MYSQL_USER: 'npm'
      MYSQL_PASSWORD: 'npm!m0stSecretPa55w0rd'
    volumes:
      - ./data/mysql:/var/lib/mysql
    networks:
      - npm-internal

networks:
  nginx:
    name: nginx
  npm-internal:
```

start with `docker-compose -p "nginx-proxy-manager" up -d`

default credentials `admin@example.com` and `changeme`

Caddy

Single Docker Compose use

Development

Code Server (Vistual Studio Code)

- https://docs.linuxserver.io/images/docker-code-server

```
docker run -d \
  --name=code-server \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Europe/Vienna \
  -e PASSWORD=password `#optional` \
  -e HASHED_PASSWORD= `#optional` \
  -e SUDO_PASSWORD=password `#optional` \
  -e SUDO_PASSWORD_HASH= `#optional` \
  -e PROXY_DOMAIN=code-server.my.domain `#optional` \
  -e DEFAULT_WORKSPACE=/config/workspace `#optional` \
  -p 8443:8443 \
  -v /path/to/appdata/config:/config \
  --restart unless-stopped \
  lscr.io/linuxserver/code-server:latest
```

- minimal cli command:

```
docker run -d \
  --name=code-server \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Europe/Vienna \
  -e PASSWORD=password \
  -e SUDO_PASSWORD=password \
  -e PROXY_DOMAIN=code.vps2.draab.at \
  -p 8443:8443 \
  -v ~/code-server/config:/config \
  --restart unless-stopped \
```

```
   lscr.io/linuxserver/code-server:latest
```

Hoppscotch Api client

https://github.com/hoppscotch/hoppscotch

GlitchTip

https://glitchtip.com/

https://glitchtip.com/assets/docker-compose.sample.yml

MySql

phpmyadmin

docker compose snippet:

```
phpmyadmin:
   image: 'phpmyadmin:latest'
   ports:
      - 8080:80
   networks:
      - <NETWORK>
   environment:
      - PMA_ARBITRARY=1
```

Mail

Mailcatcher

https://mailcatcher.me/

```
sudo docker run -d --rm -p 1025:1025 -p 1080:1080 --name mailcatcher sj26/mailcatcher
```

- `localhost:1025` config for mail server
- http://localhost:1080 to view mails tried to sent

Mailhog

Github

```
sudo docker run -d --rm -p 1025:1025 -p 8025:8025 --name mailhog mailhog/mailhog:latest
```

go to http://localhost:8025

MailPit

Homepage

remove container after it stops:

```
docker run -d \
--rm \
--name=mailpit \
-p 8025:8025 \
-p 1025:1025 \
axllent/mailpit
```

keep container running

```
docker run -d \
--restart unless-stopped \
--name=mailpit \
-p 8025:8025 \
-p 1025:1025 \
axllent/mailpit
```

docker compose:

```
service:

    mailpit:
```

```
    image: 'axllent/mailpit:latest'
    ports:
        - '${FORWARD_MAILPIT_PORT:-1025}:1025'
        - '${FORWARD_MAILPIT_DASHBOARD_PORT:-8025}:8025'
    networks:
        - <NETWORK>
```

Monitoring

Uptime Kuma

Link: https://github.com/louislam/uptime-kuma

- does not work on RPi ?
- How to install: `sudo docker run -d --restart=always -p 3021:3001 -v uptime-kuma:/app/data --name uptime-kuma louislam/uptime-kuma`
- or:

```
version: "2"
services:
  uptime-kuma:
    image: louislam/uptime-kuma
    container_name: uptime-kuma
    environment:
      - TZ=Europe/Vienna
    volumes:
      - uptime-kuma:/app/data
    ports:
      - 3021:3001
    restart: unless-stopped
volumes:
  uptime-kuma:
```

monitoring

- https://github.com/nicolargo/glances

watchtower - automatic doimage update

- https://containrrr.dev/watchtower/
- Use cases: https://www.howtogeek.com/devops/how-to-automate-docker-container-updates-with-watchtower/

Running On-Demand

Watchtower is designed to run as a long-lived daemon that continually monitors containers for updates. Sometimes you might want to manually check for new images on-demand. You can do this with the --run-once command flag:

```
docker run --rm \
    -v /var/run/docker.sock:/var/run/docker.sock \
    containrrr/watchtower --run-once
```

This will perform a single update attempt for all your running containers. The Watchtower container will then stop and be removed.

restirct to containers

you can restrict watchtower to monitoring a subset of the running containers by specifying the container names as arguments when launching watchtower.

```
docker run --rm \
    --name watchtower \
    -v /var/run/docker.sock:/var/run/docker.sock \
    containrrr/watchtower --run-once \
    portainer pihole
```

monitoring

```
docker run -d --name watchtower \
    -v /var/run/docker.sock:/var/run/docker.sock \
    containrrr/watchtower --monitor-only
```

use notification

```
docker run -d --name watchtower \
    -v /var/run/docker.sock:/var/run/docker.sock \
    containrrr/watchtower --monitor-only --notification-
url=telegram://token@telegram?chats=channel-1[,chat-id-1,...]
```

```
docker run --rm --name watchtower \
    -v /var/run/docker.sock:/var/run/docker.sock \
    containrrr/watchtower --monitor-only --run-once --notification-
url=telegram://61...85:AAHa.....mn8@telegram?chats=-922625204
```

Collaboration

Cryptpad

Link: https://hub.docker.com/r/promasu/cryptpad

Etherpad

Links:

- https://hub.docker.com/r/etherpad/etherpad
- https://github.com/ether/etherpad-lite/blob/develop/doc/docker.md

```
version: '3'

networks:
  etherpad:

services:
  db:
    restart: always
    image: "postgres:10"
    environment:
      - POSTGRES_DB=etherpad
      - POSTGRES_USER=etherpad
      - POSTGRES_PASSWORD=password
    networks:
      - etherpad
  app:
    image: "etherpad/etherpad"
    ports:
      - 9001:9001
    environment:
      - DB_TYPE=postgres
      - DB_HOST=db
```

```
      - DB_PORT=5432
      - DB_NAME=etherpad
      - DB_USER=etherpad
      - DB_PASS=secr4Ether
    networks:
      - etherpad
```

open project

https://www.openproject.org/de/docs/installation-and-operations/installation/docker/

```
docker run -d -p 8087:80 --name openproject \
  -e SERVER_HOSTNAME=your public domain \
  -e SECRET_KEY_BASE=secretKey \
  -v op_pgdata:/var/openproject/pgdata \
  -v op_assets:/var/openproject/assets \
  openproject/community:12
```

- settings:
  - Administration -> General

4minitz - meeting documentation

- Quick deployment:
  https://github.com/4minitz/4minitz/blob/develop/README.md#deployment-quick-start

```
version: '3.1'

# This is an example Docker compose configuration file
# that can be used to deploy 4Minitz via Docker.
#
# If you are deploying a fresh install of 4Minitz feel
# free to use this. If you want to upgrade an existing
# 4Minitz installation please see our migration guide.
# Otherwise you may run into issues with the MongoDB.

volumes:
  4minitz_mongodb:
  4minitz_storage:
```

```
services:

  mongo:
    image: mongo:3.4
    restart: always
    networks:
      - 4minitz
    volumes:
      - 4minitz_mongodb:/data/db

  4minitz:
    image: 4minitz/4minitz:stable
    restart: always
    ports:
      - 3100:3333
    environment:
      - MONGO_URL=mongodb://mongo/4minitz
      - ROOT_URL=http://localhost:3100
    volumes:
      - 4minitz_storage:/4minitz_storage
    networks:
      - 4minitz
    depends_on:
      - mongo

networks:
  4minitz:
```

- https://github.com/4minitz/4minitz/blob/develop/doc/admin/adminguide.md#configuration-for-sending-emails

wikiJS

https://js.wiki

```
version: "3"
services:
```

```
  db:
    image: postgres:11-alpine
    environment:
      POSTGRES_DB: wiki
      POSTGRES_PASSWORD: wikijsrocks
      POSTGRES_USER: wikijs
    logging:
      driver: "none"
    restart: unless-stopped
    volumes:
      - db-data:/var/lib/postgresql/data

  wiki:
    image: ghcr.io/requarks/wiki:2
    depends_on:
      - db
    environment:
      DB_TYPE: postgres
      DB_HOST: db
      DB_PORT: 5432
      DB_USER: wikijs
      DB_PASS: wikijsrocks
      DB_NAME: wiki
    restart: unless-stopped
    ports:
      - "3001:3000"

volumes:
  db-data:
```

joplin server

```
version: '3'

services:
    db:
        image: postgres:13
        volumes:
            - joplin-db-data:/var/lib/postgresql/data
```

```
        ports:
            - "5432:5432"
        restart: unless-stopped
        environment:
            - POSTGRES_PASSWORD=joplinPgPw
            - POSTGRES_USER=joplinPgUser
            - POSTGRES_DB=joplinPgDb
    app:
        image: joplin/server:latest
        depends_on:
            - db
        ports:
            - "22300:22300"
        restart: unless-stopped
        environment:
            - APP_PORT=22300
            - APP_BASE_URL=https://yourDomain
            - DB_CLIENT=pg
            - POSTGRES_PASSWORD=joplinPgPw
            - POSTGRES_DATABASE=joplinPgDb
            - POSTGRES_USER=joplinPgUser
            - POSTGRES_PORT=5432
            - POSTGRES_HOST=db
volumes:
    joplin-db-data:
```

Login with

email: admin@localhost password: admin

Files/Storage

Filebrowser

For loading files to the server... (like music, backup files, ...) https://filebrowser.org/installation

```
sudo \
docker run -d \
    -v /:/srv \
    -p 8012:80 \
    --name filebrowser \
    filebrowser/filebrowser
```

Default credentials: admin:admin

PSI Transfer - DataTransfer

https://github.com/psi-4ward/psitransfer

jdownloader

https://github.com/jaymoulin/docker-jdownloader

```
docker run -d --init --restart=always \
-v </path/to/downloads>:/opt/JDownloader/Downloads \
-v </path/to/appdata/config>:/opt/JDownloader/app/cfg \
--name jdownloader -u $(id -u) -p 3129:3129 \
-e MYJD_USER=email@email.com \
-e MYJD_PASSWORD=bar \
-e MYJD_DEVICE_NAME=goofy jaymoulin/jdownloader
```

Media

Ampache - Music streaming

Jellyfin

https://jellyfin.org/docs/general/installation/container

Network

wireguard

- https://jakew.me/2020/10/19/wireguard-docker/
- https://hub.docker.com/r/linuxserver/wireguard
- https://goneuland.de/wireguard-ui-wireguard-webinterface-mittels--compose-und-traefik-installieren/

```
version: "2.1"
services:
  wireguard:
    image: lscr.io/linuxserver/wireguard:latest
    container_name: wireguard
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Vienna
      - SERVERURL=home.draab.at #optional
      - SERVERPORT=51820 #optional
      - PEERS=laptop,phone #optional
      - PEERDNS=192.168.0.254 #optional
      - INTERNAL_SUBNET=10.13.13.0 #optional
#      - ALLOWEDIPS=0.0.0.0/0 #optional
      - LOG_CONFS=true #optional
    volumes:
      - wireguard-config:/config
      - /lib/modules:/lib/modules
    ports:
      - 51820:51820/udp
    sysctls:
      - net.ipv4.conf.all.src_valid_mark=1
```

```
    restart: unless-stopped
volumes:
    wireguard-config:
```

port scanner

https://github.com/JulienChapron/port-scanner-docker

wake on lan

- wolweb - wake on lan web
  - https://github.com/sameerdhoot/wolweb

PiHole

- github
- docker hub

install

via script (recommended)

https://github.com/pi-hole/docker-pi-hole/blob/master/docker_run.sh

- use the script docker_run.sh in this repo

via docker-compose

```
version: "3"

# More info at https://github.com/pi-hole/docker-pi-hole/ and https://docs.pi-hole.net/
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp"
```

```
    - "80:80/tcp"
  environment:
    TZ: 'Europe/Vienna'
    # WEBPASSWORD: 'set a secure password here or it will be random'
  # Volumes store your data between container upgrades
  volumes:
    - './etc-pihole/:/etc/pihole/'
    - './etc-dnsmasq.d/:/etc/dnsmasq.d/'
  # Recommended but not required (DHCP needs NET_ADMIN)
  #   https://github.com/pi-hole/docker-pi-hole#note-on-capabilitiess
  cap_add:
    - NET_ADMIN
  restart: unless-stopped
```

- Run `docker-compose up --detach` to build and start pi-hole

add dns wildcard domains

https://brandonrozek.com/blog/wildcarddomainspihole/

- create new file in `/etc/dnsmasq.d/03-custom-dns.conf`
- and add lines for wildcard dns entry:

```
address=/home.draab.at/192.168.0.251
address=/home.local/192.168.0.251
```

  - with command via portainer shell:

```
printf
"address=/home.draab.at/192.168.0.251\naddress=/home.local/192.168.0.251\n">/etc
/dnsmasq.d/03-custom-dns.conf
```

- exit container shell and restart

Database

MariaDB with PHPmyAdmin

- code snipped for mariadb with phpmyadmin (use only for development)

```
  mariadb:
    image: mariadb:10.6
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: YOUR_ROOT_PASSWORD_HERE
      MYSQL_USER:  YOUR_MYSQL_USER_HERE
      MYSQL_PASSWORD: YOUR_USER_PW_HERE
    ports:
      - "3306:3306"
    volumes:
      - mariadb:/var/lib/mysql
    networks:
      db:

  phpmyadmin:
    image: phpmyadmin
    restart: always
    ports:
      - "8080:80"
    environment:
      - PMA_HOST=mariadb
      - PMA_PORT=3306
    networks:
      db:
```

Mongo db

docker web client

github

```
docker run -d --name mongoku -p 3100:3100 --env MONGOKU_DEFAULT_HOST="mongodb://mongo"
huggingface/mongoku
```

opensearch

one liner

```
docker run -d -p 9200:9200 -p 9600:9600 --name=opensearch --restart=unless-stopped -e
"discovery.type=single-node" -e "OPENSEARCH_INITIAL_ADMIN_PASSWORD=Fronius123" -e
"DISABLE_SECURITY_PLUGIN=true" opensearchproject/opensearch:latest
```

single node with dashboard

```
services:
  opensearch-node1:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node1
    environment:
      - cluster.name=opensearch-cluster # Name the cluster
      - node.name=opensearch-node1 # Name the node that will run in this container
  - discovery.type=single-node
#      - discovery.seed_hosts=opensearch-node1,opensearch-node2 # Nodes to look for when
discovering the cluster
#      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2 # Nodes
eligibile to serve as cluster manager
      - bootstrap.memory_lock=true # Disable JVM heap memory swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least
50% of system RAM
      - "DISABLE_INSTALL_DEMO_CONFIG=true" # Prevents execution of bundled demo script which
installs demo certificates and security configurations to OpenSearch
      - "DISABLE_SECURITY_PLUGIN=true" # Disables Security plugin
```

```
    ulimits:
      memlock:
        soft: -1 # Set memlock to unlimited (no soft or hard limit)
        hard: -1
      nofile:
        soft: 65536 # Maximum number of open files for the opensearch user - set to at least
65536
        hard: 65536
    volumes:
      - opensearch-data1:/usr/share/opensearch/data # Creates volume called opensearch-data1
and mounts it to the container
    ports:
      - 9200:9200 # REST API
      - 9600:9600 # Performance Analyzer
    networks:
      - opensearch-net # All of the containers will join the same Docker bridge network
□
  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:latest
    container_name: opensearch-dashboards
    ports:
      - 5601:5601 # Map host port 5601 to container port 5601
    expose:
      - "5601" # Expose port 5601 for web access to OpenSearch Dashboards
    environment:
      - 'OPENSEARCH_HOSTS=["http://opensearch-node1:9200"]'
      - "DISABLE_SECURITY_DASHBOARDS_PLUGIN=true" # disables security dashboards plugin in
OpenSearch Dashboards
    networks:
      - opensearch-net

volumes:
  opensearch-data1:

networks:
  opensearch-net:
```

multi node

https://docs.opensearch.org/latest/install-and-configure/install-opensearch/docker/#sample-docker-compose-file-for-development

```
services:
  opensearch-node1:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node1
    environment:
      - cluster.name=opensearch-cluster # Name the cluster
      - node.name=opensearch-node1 # Name the node that will run in this container
      - discovery.seed_hosts=opensearch-node1,opensearch-node2 # Nodes to look for when discovering the cluster
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2 # Nodes eligibile to serve as cluster manager
      - bootstrap.memory_lock=true # Disable JVM heap memory swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least 50% of system RAM
      - "DISABLE_INSTALL_DEMO_CONFIG=true" # Prevents execution of bundled demo script which installs demo certificates and security configurations to OpenSearch
      - "DISABLE_SECURITY_PLUGIN=true" # Disables Security plugin
    ulimits:
      memlock:
        soft: -1 # Set memlock to unlimited (no soft or hard limit)
        hard: -1
      nofile:
        soft: 65536 # Maximum number of open files for the opensearch user - set to at least 65536
        hard: 65536
    volumes:
      - opensearch-data1:/usr/share/opensearch/data # Creates volume called opensearch-data1 and mounts it to the container
    ports:
      - 9200:9200 # REST API
      - 9600:9600 # Performance Analyzer
    networks:
      - opensearch-net # All of the containers will join the same Docker bridge network
  opensearch-node2:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node2
    environment:
```

```
      - cluster.name=opensearch-cluster # Name the cluster
      - node.name=opensearch-node2 # Name the node that will run in this container
      - discovery.seed_hosts=opensearch-node1,opensearch-node2 # Nodes to look for when
discovering the cluster
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2 # Nodes
eligibile to serve as cluster manager
      - bootstrap.memory_lock=true # Disable JVM heap memory swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least
50% of system RAM
      - "DISABLE_INSTALL_DEMO_CONFIG=true" # Prevents execution of bundled demo script which
installs demo certificates and security configurations to OpenSearch
      - "DISABLE_SECURITY_PLUGIN=true" # Disables Security plugin
    ulimits:
      memlock:
        soft: -1 # Set memlock to unlimited (no soft or hard limit)
        hard: -1
      nofile:
        soft: 65536 # Maximum number of open files for the opensearch user - set to at least
65536
        hard: 65536
    volumes:
      - opensearch-data2:/usr/share/opensearch/data # Creates volume called opensearch-data2
and mounts it to the container
    networks:
      - opensearch-net # All of the containers will join the same Docker bridge network
  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:latest
    container_name: opensearch-dashboards
    ports:
      - 5601:5601 # Map host port 5601 to container port 5601
    expose:
      - "5601" # Expose port 5601 for web access to OpenSearch Dashboards
    environment:
      - 'OPENSEARCH_HOSTS=["http://opensearch-node1:9200","http://opensearch-node2:9200"]'
      - "DISABLE_SECURITY_DASHBOARDS_PLUGIN=true" # disables security dashboards plugin in
OpenSearch Dashboards
    networks:
      - opensearch-net

volumes:
```

```
  opensearch-data1:
  opensearch-data2:

networks:
  opensearch-net:
```

Games

minecraft

java edition

```
version: "3.8"

services:
  mc:
    image: itzg/minecraft-server
    tty: true
    stdin_open: true
    ports:
      - "25565:25565"
    environment:
      EULA: "TRUE"
      MEMORY: 1G
      SERVER_NAME: "DRaab Minecraft Server"
    volumes:
      # attach the relative directory 'data' to the container's /data path
      - minecraftData:/data
volumes:
  minecraftData:
```

bedrock version

necessary for android

- https://github.com/itzg/docker-minecraft-bedrock-server

```
version: '3.4'

services:
  bds:
    image: itzg/minecraft-bedrock-server
    environment:
      EULA: "TRUE"
      GAMEMODE: survival
      DIFFICULTY: normal
      ALLOW_LIST_USERS: "Antiker90"
    ports:
      - "19132:19132/udp"
    volumes:
      - bds:/data
    stdin_open: true
    tty: true

volumes:
  bds: {}
```

server props

gamemode: Allowed values: "survival", "creative", or "adventure"

Finance

i hate money

https://ihatemoney.readthedocs.io/en/latest/

facto

https://github.com/nymanjens/facto

Projects

Home Automation

mqtt

client

```
docker run -d --rm --network=nodered_node-red-net --network=npm_default --name mqttx-web -p 8000:80 emqx/mqttx-web
```

Dockerfile

Dockerfile

- dockerfile manuals:
  - https://takacsmark.com/dockerfile-tutorial-by-example-dockerfile-best-practices-2018/
- best practice:
  - https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

key instructions

- `FROM` - base image
- `ENV variable=test`
  - environment variable
  - can be used with bash syntax `$variable`
- `VOLUME [ "/var/test" ]`
  - used to expose some storage
- `CMD`
  - what command should be run in the image
  - `CMD [“executable”, “param1”, “param2”, ...]`
  - Its possible to override the command on starting up a container `docker run myImage:latest _command_`
  - Only last `CMD` will be used (so specifiy only one).
  - Its a good practice to specify at leaset a shell command here.
- `ENTRYPOINT`
  - Used to specify the 'main executable' of the image
  - If `ENTRYPOINT` is specified, `CMD` paramas will be added to the `ENTRYPOINT`
  - ```
    ENTRYPOINT ["git"]
    CMD ["--help"]
    ```

building

Docker CLI
    ```

Docker-compose CLI

Basics

use `docker-compose -p "nginx-proxy-manager" up -d` to:

- `-p` with custom stack name
- `-d` as deamon (in background)
- `-f` from specific docker-compose file

docker compose file

- https://docs.docker.com/compose/compose-file/compose-file-v3/

```
version: '3.1'

services:


  my-car:

    image: my-car-next-js
    user: 1000:1000
    build:
      context: ./
      dockerfile: "./development.Dockerfile"
    ports:
      - 3000:3000
    volumes:
      - ./:/app
    entrypoint: ["sh", "./development.entrypoint.sh"]
```

Podman

install

docs

- is deamonless
- uses the same cli specificatio than docker
- `docker run ...` -> `podman run ...`
- `docker-compose ...` -> `podman-compose ...` must be installed before

autostart at boot

linuxhandbook

podman is deamonless so it wont autostart, create a sytem entry with:

```
podman generate systemd --new --name CONTAINER_NAME -f /home/USER/CONTAINER_NAME.service
mv /home/USER/CONTAINER_NAME.service /home/USER/.config/systemd/user/CONTAINER_NAME.service

# for root
sudo systemctl daemon-reload
sudo systemctl enable SERVICE_NAME.service
sudo systemctl status SERVICE_NAME.service

# or other user
systemctl --user daemon-reload
systemctl --user enable SERVICE_NAME.service
systemctl --user status SERVICE_NAME.service
```