Docker

- environment
- pihole with http via traefik 2.0
- samba
- sabnzbd
- docker installation
- cleanup
- traefik 2.0
- authelia
- kubernetes
- docker-compose
- nextcloud migration
- matomo
- graylog

environment

The environment file can be used to store global configuration parameters for your docker-compose file. The file is stored in `/etc/environment`.

The following configuration is largely based on [docker media server how to].

```
vim /etc/environment
```

After a standard ubuntu 18.04 installation the environment file probably looks like this

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
"
```

User and Group

In my basic setup I create a user `cloud` and a group `cloud` that I run docker-compose with, in my environment file I specify the ids of the user and group as parameters.

```
PUID=1002
PGID=1002
```

You can get the values for your user via the following command

```
sudo id cloud
```

this shoudl output some thing like this

```
uid=1002(cloud) gid=1002(cloud) groups=1002(cloud)
```

Time zone

You can add a valiable for your timezone [time zones]

```
TZ="Europe/Zurich"
```

Basic auth

You can specify a basic auth user that you can use to secure pages, you can encode the password as follows

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

You might need to install some extra dependencies for the apache tools

```
apt install apache2-utils -y
```

That would result in some thing as `admin:$$apr1$$o6BgYlgS$$U3GfcrYe6/7Ir2bVvVit61` that you could than add as follows

```
HTTP_USERNAME=admin
HTTP_PASSWORD="$$apr1$$o6Bgbodo@naumann.devYlgS$$U3GfcrYe6/7Ir2bVvVit61"
```

Clear text password

Some configurations need clear text passwords

```
U01_CLEAR=bodo
U02_CLEAR=tv
GRP_CLEAR=cloud
P01_CLEAR=supersecret
P02_CLEAR=supersecret
```

Domain name

You can configure your host name as domain name and reference it in all services, this makes it easy to reuse a docker file on different hosts.

```
DOMAINNAME="example.com"
```

DNS Server

The default DNS Server

```
DNS=192.168.1.254
```

email address

```
EMAIL=mail@example.com
```

Sample

This is a sample based on the parameters above

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
"
PUID=1002
PGID=1002
TZ="Europe/Zurich"
HTTP_USERNAME=admin
HTTP_PASSWORD="$$apr1$$o6Bgbodo@naumann.devYlgS$$U3GfcrYe6/7Ir2bVvVit61"
U01_CLEAR=bodo
U02_CLEAR=tv
GRP_CLEAR=cloud
P01_CLEAR=supersecret
P02_CLEAR=supersecret
DOMAINNAME="example.com"
DNS=1.1.1.1
EMAIL=mail@example.com
```

pihole with http via traefik 2.0

This docker-compose file shows a basic traefik 2.0. configuration.

```
version: '3.7'

services:
  #
  traefik:
    container_name: traefik
    domainname: ${DOMAINNAME}
    image: traefik
    restart: unless-stopped
    command:
      - --api.insecure=true
      - --providers.docker
      - --providers.docker.exposedbydefault=false
      - --entrypoints.web.address=:80
      #- --providers.docker.defaultRule="Host(`${DOMAINNAME}`)"
    ports:
      - "80:80"
      - "443:443"
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - default
      - discovery
    dns:
      - ${DNS}

  pihole:
    container_name: pihole
    domainname: ${DOMAINNAME}
```

```
    image: pihole/pihole:latest
    dns:
      - 127.0.0.1
      - 192.168.1.254
    ports:
      - '0.0.0.0:53:53/tcp'
      - '0.0.0.0:53:53/udp'
      - '0.0.0.0:67:67/udp'
    volumes:
      - pihole:/etc/pihole/
      - dnsmasq:/etc/dnsmasq.d/
    environment:
      VIRTUAL_HOST: ${DOMAINNAME}
      TZ: ${TZ}
      WEBPASSWORD: 'usergo'
      DNS1: 192.168.1.254
      DNS2: 'no'
      DNSSEC: 'False'
      DNS_BOGUS_PRIV: 'True'
      CONDITIONAL_FORWARDING: 'True'
      CONDITIONAL_FORWARDING_IP: 192.168.1.254
      CONDITIONAL_FORWARDING_DOMAIN: 'home'
    restart: unless-stopped
    labels:
       - "traefik.enable=true"
       - "traefik.http.routers.pihole.rule=Host(`${DOMAINNAME}`)"
       - "traefik.http.routers.pihole.entrypoints=web"
       - "traefik.http.services.pihole.loadbalancer.server.port=80"


networks:
  discovery:
volumes:
  pihole:
  dnsmasq:
```

samba

Create the first `docker-compose.yml` file

```
su - cloud
```

Switch to `bash` as shell

```
bash
```

Create a docker folder

```
mkdir /home/cloud/docker
```

Open the docker file

```
vim /home/cloud/docker/docker-compose.yml
```

docker-compose.yml

This is a basic samba configuration that you can use to acces your shared volumes via smb protocol

```
version: '3.7'
services:
  samba:
    image: dperson/samba
    environment:
      - TZ=${TZ}
      - WORKGROUP=WORKGROUP
      - USER=${U01_CLEAR};${P01_CLEAR}
      - USERID=${PUID}
      - GROUPID=${PGID}
    networks:
      - default
    ports:
      - '137:137/udp'
```

```
      - '138:138/udp'
      - '139:139/tcp'
      - '445:445/tcp'
    read_only: true
    tmpfs:
      - /tmp
    restart: unless-stopped
    stdin_open: true
    tty: true
    volumes:
      - samba:/mnt:z
    command:
      -u "${U02_CLEAR};${P02_CLEAR}"
      -s "tv;/mnt/tv;no;no;no;${U02_CLEAR}"
      -s "share;/mnt/share;yes;no;yes;all"
      -s "data;/mnt/data;no;no;no;${U01_CLEAR}"
      -p

volumes:
  samba:
```

Start the container

Start samba with docker compose for the first time

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

```
cloud@itx02:~/docker$ docker-compose up -d
Creating network "docker_default" with the default driver
Creating volume "docker_samba" with default driver
Pulling samba (dperson/samba:)...
latest: Pulling from dperson/samba
89d9c30c1d48: Pull complete
7ab5ece07d0a: Pull complete
95fd3ce218b9: Pull complete
Digest: sha256:ffbca71bee5396195df4987acd87dca4fbb66906a9888783193fe57ca9854acc
Status: Downloaded newer image for dperson/samba:latest
Creating docker_samba_1 ... done
cloud@itx02:~/docker$ docker-compose ps
```

```
     Name                   Command
State
Ports
-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------
docker_samba_1   /sbin/tini -- /usr/bin/sam ...   Up (health: starting)   0.0.0.0:137-
>137/udp, 0.0.0.0:138->138/udp, 0.0.0.0:139->139/tcp, 0.0.0.0:445->445/tcp
```

sabnzbd

```
version: '3.6'

services:
  traefik:
    container_name: traefik
    domainname: ${DOMAINNAME}
    image: traefik
    restart: unless-stopped
    command: --api.insecure=true --providers.docker --providers.docker.exposedbydefault=false
--entrypoints.web.address=:80 #--providers.docker.defaultRule="Host(`${DOMAINNAME}`)"
    ports:
      - "80:80"
      - "443:443"
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - default
      - discovery
    dns:
      - ${DNS}

  sabnzbd:
    image: 'linuxserver/sabnzbd'
    container_name: 'sabnzbd'
    volumes:
      - ./sabnzbd/sabnzbd.ini:/config/sabnzbd.ini
      - sabnzbd-config:/config
      - samba:/samba
    restart: always
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    networks:
```

```
      - discovery
    depends_on:
      - traefik
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.sabnzbd.rule=Host(`sabnzbd.${DOMAINNAME}`)"
      - "traefik.http.routers.sabnzbd.entrypoints=web"
      - "traefik.http.services.sabnzbd.loadbalancer.server.port=8080"

  samba:
    image: dperson/samba
    environment:
      - TZ=${TZ}
      - WORKGROUP=WORKGROUP
      - USER=bodo;supersecret
      - USERID=${PUID}
      - GROUPID=${PGID}
      - PERMISSIONS
    networks:
      - default
    ports:
      - '137:137/udp'
      - '138:138/udp'
      - '139:139/tcp'
      - '445:445/tcp'
    read_only: true
    tmpfs:
      - /tmp
    restart: unless-stopped
    stdin_open: true
    tty: true
    volumes:
      - samba:/mnt:z
    command:
      -s "tmp;/mnt/tmp;no;no;no;bodo"
      -s "downloads;/mnt/downloads;no;no;no;bodo"
      -s "watch;/mnt/watch;no;no;no;bodo"

volumes:
  sabnzbd-config:
```

```
  samba:
networks:
  discovery:
  default:
    driver: bridge
```

docker installation

This documentation shows how to install docker from the docker ppa

Create users and groups

First create a user and group `cloud`

```
useradd -u 1002 -m cloud
```

Add the user `cloud` to the sudo group

```
usermod -a -G sudo cloud
```

Add a `docker` group

```
groupadd docker
```

Add the user `cloud` to the docker group

```
usermod -aG docker cloud
```

Gibe the user the bash shell

```
usermod --shell /bin/bash cloud
```

basic dependencies

Ubuntu 22.04 LTS

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Ubuntu 24.04 LTS

```
apt install apt-transport-https curl
```

Install some basic dependencies

```
apt update && apt upgrade -y && apt install vim screen lsof apt-transport-https ca-
certificates curl software-properties-common -y
```

Ubuntu 20.04

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-
common
```

docker repository configuration

Ubuntu 24.04 LTS

```
https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME")
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
apt update
```

Add the repository key of the docker repository

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Now add the apt configuration for the repository

```
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

You could also alternatively add the `testing` repository instead of the `stable` repository

```
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) test"
```

install docker

```
apt update && apt install docker-ce docker-ce-cli containerd.io -y
```

install docker-compose

Get the latest docker-compose

```
curl -L  https://github.com/docker/compose/releases/download/v2.27.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
```

Set the proper rights

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

And check if `docker-compose` works

```
docker-compose --version
```

This should give a response like this

```
docker-compose version 1.25.0-rc2, build 661ac20e
```

cleanup

Some times you want to clean up your docker environment after experimenting, this shows you how you can do it.

Prune images

```
docker image prune -a
```

Prune containers

```
docker container prune
```

Prune volumes

```
docker volume prune
```

Prune networks

```
docker network prune
```

Prune everything

```
docker system prune --volumes
```

Stop the container(s) using the following command:

```
docker-compose down
```

Delete all containers using the following command:

```
docker rm -f $(docker ps -a -q)
```

Delete all volumes using the following command:

```
docker volume rm $(docker volume ls -q)
```

Restart the containers using the following command:

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

traefik 2.0

Into how toos part 1 and part 2

authelia

The following page documents how I did setup a service in docker-compose to use authelia for authentication via traefik 2.0

environment

I use the following entries for this setup in my `/etc/environment` file

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
"
PUID=1000
PGID=1000
TZ="Europe/Zurich"
DOMAINNAME="example.com"
DNS=1.1.1.1
GOPATH=/usr/bin/go
EMAIL=mail@example.com
```

Install golang

I found a setup guide that shows how to install golang on ubuntu 18.04 based on a ppa. I did the following steps

```
sudo add-apt-repository ppa:longsleep/golang-backports
sudo apt-get update
sudo apt-get install golang-go
```

Basic traefik 2.0 setup

My basic traefik 2.0 setup was based on the traefik 2.0 intoduction blog post. While configuring I just stumble upon one issue.

Full docker-compose

```
version: '3.7'

services:
  traefik:
    container_name: traefik
    domainname: ${DOMAINNAME}
    image: traefik
    restart: unless-stopped
    command:
      - --api.insecure=true
      - --providers.docker=true
      - --providers.docker.exposedbydefault=false
      - --entrypoints.web.address=:80
      - --log.level=DEBUG
      - --entrypoints.websecure.address=:443
      - --certificatesresolvers.le.acme.email=${EMAIL}
      - --certificatesresolvers.le.acme.storage=/acme.json
      - --certificatesresolvers.le.acme.tlschallenge=true
    ports:
      - "80:80"
      - "443:443"
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./users:/users
    networks:
      - default
      - discovery
    dns:
      - ${DNS}

  my-app:
    image: containous/whoami:v1.3.0
    command:
      - --port=8082
    networks:
      - discovery
    labels:
      - "traefik.enable=true"
```

```
      - "traefik.http.routers.my-app.rule=Host(`my-app.${DOMAINNAME}`)"
      - "traefik.http.services.my-app.loadbalancer.server.port=8082"
      - "traefik.http.routers.my-app.middlewares=authme"
      - "traefik.http.middlewares.authme.forwardauth.address=http://authelia:9091"
      - "traefik.http.middlewares.authme.forwardauth.trustforwardheader=true"
      - "traefik.http.middlewares.authme.forwardauth.authresponseheaders=X-Forwarded-User"
      -
"traefik.http.middlewares.authme.forwardauth.address=http://authelia:8080/api/verify?rd=https:
//auth.${DOMAINNAME}/%23/"
      - "traefik.http.routers.my-app.tls.certresolver=le"
      - "traefik.http.routers.my-app.entrypoints=websecure"

  authelia:
    image: clems4ever/authelia:master
    container_name: authelia
    restart: always
    volumes:
      - ./authelia/config.minimal.yml:/etc/authelia/config.yml:ro
      - ./authelia/users_database.yml:/etc/authelia/users_database.yml:rw
      - authelia:/tmp/authelia
      - ${GOPATH}:/go
    environment:
      - TZ=${TZ}
      - NODE_TLS_REJECT_UNAUTHORIZED=1
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.auth.rule=Host(`auth.${DOMAINNAME}`)"
      - "traefik.http.routers.auth.entrypoints=web"
      - "traefik.http.services.auth.loadbalancer.server.port=8080"
      - "traefik.http.routers.auth.tls.certresolver=le"
      - "traefik.http.routers.auth.entrypoints=websecure"
    expose:
      - 8080
    networks:
      - discovery

volumes:
  authelia:
networks:
  discovery:
```

authelia config

This is the `users_database.yml` sample that contains a user `testuser` with password `test`

```
users:
  testuser: ## I have set the password below to 'test' for you
    password:
'{CRYPT}$6$rounds=500000$Bui4ldW5hXOI9qwJ$IUHQPCusUKpTs/OrfE9UuGb1Giqaa5OZA.mqIpH.Hh8RGFsEBHVi
CwQDx6DfkGUiF60pqNubFBugfTvCJIDNw1'
    email: your@email.address
    groups:
      - admins
      - dev
```

This is my `config.minimal.yml` for this sample, its all base on a working sample for traefik that I found googeling.

```
###############################################################
#                Authelia minimal configuration               #
###############################################################

#logs_level: debug

# The secret used to generate JWT tokens when validating user identity by
# email confirmation.
jwt_secret: supersecret

authentication_backend:
  file:
    path: /etc/authelia/users_database.yml

session:
  secret: change_this_for_your_server
  domain: personal.domain

# Configuration of the storage backend used to store data and secrets. i.e. totp data
storage:
  local:
    path: /etc/authelia/storage
```

```
# TOTP Issuer Name
#
# This will be the issuer name displayed in Google Authenticator
# See: https://github.com/google/google-authenticator/wiki/Key-Uri-Format for more info on
issuer names
totp:
  issuer: personal.domain

# Access Control
#
# Access control is a set of rules you can use to restrict user access to certain
# resources.
access_control:
  # Default policy can either be `bypass`, `one_factor`, `two_factor` or `deny`.
  default_policy: one_factor

  rules:
    - domain: public.personal.domain
      policy: bypass
    - domain: httpbin.personal.domain
      policy: bypass
    - domain: auth.cusack.cloud
      policy: bypass
    - domain: firewall.personal.domain
      policy: two_factor
    - domain: proxmox.personal.domain
      policy: two_factor
#      resources:
#        - '^/api/.*$'
#        - '^/notifications/.*$'
      policy: bypass

#    - domain: who.example.com
#      policy: two_factor

# Configuration of the authentication regulation mechanism.
regulation:
  # Set it to 0 to disable max_retries.
  max_retries: 5
```

```
  # The user is banned if the authenticaction failed `max_retries` times in a `find_time`
seconds window.
  find_time: 120

  # The length of time before a banned user can login again.
  ban_time: 180

# Configuration of session cookies
#
# The session cookies identify the user once logged in.
session:
  # The name of the session cookie. (default: authelia_session).
  name: authelia_session

  # The secret to encrypt the session cookie.
  secret: change_this_for_your_server

  # The time in ms before the cookie expires and session is reset.
  expiration: 604800000 # 1 week

  # The inactivity time in ms before the session is reset.
  inactivity: 300000 # 5 minutes

  # The domain to protect.
  # Note: the authenticator must also be in that domain. If empty, the cookie
  # is restricted to the subdomain on the issuer.
  domain: personal.domain



# Default redirection URL
#
# Note: this parameter is optional. If not provided, user won't
# be redirected upon successful authentication.
#default_redirection_url: https://authelia.example.domain

#notifier:
  # For testing purpose, notifications can be sent in a file
#  filesystem:
#    filename: /tmp/authelia/notification.txt
```

```
notifier:
  smtp:
#    username:
#    password:
    secure: false
    host: mail
    port: 25
    sender: docker@your-mail-server
```

kubernetes

```
# Setup daemon.
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

```
mkdir -p /etc/systemd/system/docker.service.d
```

```
systemctl daemon-reload
```

```
systemctl restart docker
```

```
systemctl enable docker
```

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
```

```
apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

```
apt install kubeadm -y
```

```
kubeadm version
```

```
swapoff -a
```

```
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The
recommended driver is "systemd". Please follow the guide at
https://kubernetes.io/docs/setup/cri/
[WARNING SystemVerification]: this Docker version is not on the list of validated versions:
```

```
19.03.5. Latest validated version: 18.09
```

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.80.0.3:6443 --token z2thcd.6703vccqh3nh3cb7 \
    --discovery-token-ca-cert-hash
sha256:7205d50008d1a75e005c57bc3df4a56f2135a0e85e7ef9d6e31987cc63bc7d05
```

kubeadm init --pod-network-cidr=10.80.0.0/24

```
cloud@apu03:~$ mkdir -p $HOME/.kube
cloud@apu03:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[sudo] password for cloud:
Sorry, try again.
[sudo] password for cloud:
Sorry, try again.
[sudo] password for cloud:
sudo: 2 incorrect password attempts
cloud@apu03:~$ passwd
Changing password for cloud.
(current) UNIX password:
passwd: Authentication token manipulation error
passwd: password unchanged
cloud@apu03:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[sudo] password for cloud:
cloud@apu03:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
cloud@apu03:~$ kubectl get nodes
NAME    STATUS     ROLES    AGE     VERSION
```

```
apu03   NotReady   master   3m16s   v1.16.3
```

Install Dashboard

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta6/aio/deploy/recommended.yaml
```

```
kubectl proxy
```

docker-compose

This is a collection of handy docker-compose commands.

Samba

```
docker-compose exec samba chown -R 1002:1002 /mnt
```

```
docker-compose exec samba find /mnt/ -type d -exec chmod 750 {} \;
```

```
docker-compose exec samba find /mnt/ -type f -exec chmod 640 {} \;
```

```
docker-compose exec samba chown -R 1002:1002 /syncthing
```

Nextcloud

```
docker-compose build --pull
```

```
docker-compose exec app chown -R www-data:www-data /var/www/html/
```

```
docker-compose exec app find /var/www/html/ -type d -exec chmod 750 {} \;
```

```
docker-compose exec app find /var/www/html/ -type f -exec chmod 640 {} \;
```

```
docker-compose exec --user www-data app php -d memory_limit=2048M occ upgrade
```

```
docker-compose exec --user www-data app php -d memory_limit=2048M occ files:scan --all
```

```
docker-compose exec --user www-data app php -d memory_limit=2048M occ maintenance:mode --off
```

iobroker

```
docker-compose exec iobroker chown -R 1002:1002 /opt/iobroker
```

mosquitto

```
docker-compose exec mosquitto chown -R 1002:1002 /mosquitto
```

```
docker-compose exec sabnzbd chown -R 1000:1000 /config

docker-compose exec sonarr chown -R 1000:1000 /config

docker-compose exec radarr chown -R 1000:1000 /config

docker-compose exec lazylibrarian chown -R 1000:1000 /config

docker-compose exec sabnzbd chown -R 1000:1000 /completed

docker-compose exec lazylibrarian chown -R 1000:1000 /downloads
```

nextcloud migration

Set system into maintenace mode

```
docker-compose exec --user www-data app php -d memory_limit=2048M occ maintenance:mode --on
```

Backup DB

```
docker exec db mysqldump --single-transaction -h localhost -u root -p[password] nextcloud > nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
```

Sync folders

```
rsync -Aavx data -e ssh bodo@super71.home:/var/lib/docker/data/
```

```
rsync -Aavx custom_apps -e ssh bodo@super71.home:/var/lib/docker/data/
```

```
rsync -Aavx themes -e ssh bodo@super71.home:/var/lib/docker/data/
```

```
rsync -Aavx config -e ssh bodo@super71.home:/var/lib/docker/data/
```

Drop DB

```
docker exec mariadb mysql -h localhost -u root -p[password] -e "DROP DATABASE nextcloud"
```

```
docker exec mariadb mysql -h localhost -u root -p[password] -e "CREATE DATABASE nextcloud"
```

Restore DB

```
cat nextcloud-sqlbkp_20201110.bak | docker exec -i mariadb /usr/bin/mysql -u root --password=[password] nextcloud
```

Restore Files

```
rsync -Aax /var/lib/docker/data/data/ /var/lib/docker/volumes/docker_nextcloud-data/_data/
```

```
rsync -Aax /var/lib/docker/data/custom_apps/ /var/lib/docker/volumes/docker_nextcloud-custom_apps/_data/
```

```
rsync -Aax /var/lib/docker/data/themes/ /var/lib/docker/volumes/docker_nextcloud-config/_data/www/nextcloud/themes/
```

Set rights

```
docker-compose exec nextcloud chown -R 1002:1002 /config
```

```
docker-compose exec nextcloud chown -R 1002:1002 /data
```

```
docker-compose exec nextcloud chown -R 1002:1002 /config/www/nextcloud/custom_apps
```

```
docker-compose exec nextcloud updater.phar
```

```
docker-compose exec nextcloud occ maintenance:mode --on
```

```
docker-compose exec nextcloud find /data/ -type d -exec chmod 750 {} \;
```

```
docker-compose exec nextcloud find /data/ -type f -exec chmod 640 {} \;
```

```
docker-compose exec nextcloud occ files:scan --all
```

```
docker-compose exec nextcloud occ maintenance:repair
```

```
docker-compose exec nextcloud occ db:add-missing-indices
```

```
docker-compose exec nextcloud occ db:convert-filecache-bigint
```

```
docker-compose exec nextcloud occ background-job:execute OCA\Recognize\BackgroundJobs\ClassifyImagenetJob
```

```
docker-compose exec documentserver /var/www/onlyoffice/documentserver/npm/json -f /etc/onlyoffice/documentserver/local.json 'services.CoAuthoring.secret.session.string'
```

matomo

```
docker-compose exec matomo chown -R www-data:www-data /var/www/html
docker-compose exec matomo find /var/www/html -type f -exec chmod 644 {} \;
docker-compose exec matomo find /var/www/html -type d -exec chmod 755 {} \;
docker-compose exec matomo chmod 755 /var/www/html/console
```

```
docker-compose exec matomo php /var/www/html/console core:update
```

```
docker-compose exec matomo chown -R www-data:www-data /var/www/html
```

graylog

Create users and groups

First create a user and group `graylog`

```
useradd -u 1100 -m graylog
```

Add the user `graylog` to the sudo group

```
usermod -a -G sudo graylog
```

Add the user `graylog` to the docker group

```
usermod -aG docker graylog
```

Give the user the bash shell

```
usermod --shell /bin/bash graylog
```