

---

---

***THIRTYSOMETHING***

---

---

## **DIY-NAS**

About my DIY-NAS

ThirtySomething

21.05.2022

# Contents

<b>Contents</b> . . . . .	<b>1</b>
<b>Change history</b> . . . . .	<b>3</b>
<b>1 Initial situation</b> . . . . .	<b>4</b>
<b>2 Search for alternatives</b> . . . . .	<b>5</b>
2.1 Use a mini PC in front of the NAS . . . . .	5
2.2 Hardware alternatives . . . . .	5
2.3 The planned hardware . . . . .	5
2.4 The bought hardware . . . . .	6
<b>3 The hardware build</b> . . . . .	<b>7</b>
<b>4 The software installation</b> . . . . .	<b>8</b>
4.1 The operating system . . . . .	8
4.2 The disks . . . . .	8
4.3 The RAID storage . . . . .	9
4.4 The file system . . . . .	10
4.5 Shared folders . . . . .	10
4.6 Users . . . . .	11
4.7 CIFS shares . . . . .	12
4.8 Tuning . . . . .	13
4.8.1 SMB/CIFS . . . . .	13
<b>5 OMV plugins</b> . . . . .	<b>14</b>
5.1 ClamAV . . . . .	14
5.2 MiniDLNA . . . . .	16
5.3 OMV extras . . . . .	17
5.4 Autoshutdown . . . . .	18
5.5 PhotoPrism . . . . .	20
<b>6 Docker</b> . . . . .	<b>23</b>
6.1 Docker installation . . . . .	23
6.2 Portainer . . . . .	23
6.3 Container list . . . . .	25
6.4 Pi-hole . . . . .	25
6.5 SCM-Manager . . . . .	25
6.6 Syncthing . . . . .	27

---

6.7 Watchtower . . . . .	28
6.8 MariaDB und phpMyAdmin . . . . .	29
<b>7 Ports . . . . .</b>	<b>31</b>
<b>A svnExport.sh . . . . .</b>	<b>33</b>
<b>List of Figures . . . . .</b>	<b>I</b>
<b>List of Tables . . . . .</b>	<b>III</b>
<b>Glossary . . . . .</b>	<b>IV</b>

## Change history

Version	Date	Description	Name
1.0.0	02.04.2022	<ul style="list-style-type: none"><li>Start with description</li></ul>	ThirtySomething
1.0.1	16.04.2022	<ul style="list-style-type: none"><li>Rename section WOL to Autoshutdown</li><li>Fill section Autoshutdown with content</li></ul>	ThirtySomething
1.0.2	16.04.2022	<ul style="list-style-type: none"><li>Split file DIY-NAS-Content into separate files</li></ul>	ThirtySomething
1.0.3	21.04.2022	<ul style="list-style-type: none"><li>Update to latest template version</li><li>Add section about used ports</li><li>Add svnExport.sh as appendix</li></ul>	ThirtySomething
1.0.4	23.04.2022	<ul style="list-style-type: none"><li>Add documentation about Syncing docker container</li></ul>	ThirtySomething
1.0.5	25.04.2022	<ul style="list-style-type: none"><li>Split section OMV plugins into files</li><li>Add description of PhotoPrism</li></ul>	ThirtySomething
1.0.6	28.04.2022	<ul style="list-style-type: none"><li>Add list of containers from old document</li></ul>	ThirtySomething
1.0.7	30.04.2022	<ul style="list-style-type: none"><li>Use latest template version</li><li>Add section about Pi-hole</li></ul>	ThirtySomething
1.0.8	11.05.2022	<ul style="list-style-type: none"><li>Use latest template version</li><li>Add copyright hint to the images</li></ul>	ThirtySomething
1.0.9	19.05.2022	<ul style="list-style-type: none"><li>Add description of Watchtower</li></ul>	ThirtySomething
1.0.10	21.05.2022	<ul style="list-style-type: none"><li>Add description of MariaDB and phpMyAdmin</li></ul>	ThirtySomething

**Table 1:** Change history

## 1 Initial situation

In 2010 I bought my first [NAS](#). It was a [Synology DS411slim](#). The device is running up to now. For a few years now, it is only supplied with security updates. That is quite remarkable for the fact that it has already 12 years on the hump.

Now I've run out of space - I have 4\*2TB running there in [RAID 10](#), which results in about 3.7TB. In addition, the performance is, well, not quite up to date.

## 2 Search for alternatives

### 2.1 Use a mini PC in front of the NAS

Realizing this situation I was searching for alternatives. The first idea was to use a [mini PC](#) in front of the [NAS](#). This failed for some reasons.

- Using the [NAS](#) directly for [Docker](#) volumes
- The CPU of the mini PC does not support [Intel VT](#)
- The network as bottleneck for [Docker](#) containers

### 2.2 Hardware alternatives

Then I decided to by a new [NAS](#) system. But which kind of [NAS](#) will it be? I've spent a long time to search the internet for possible soultions. I didn't know about the possibility to pimp a [TerraMaster NAS](#) with a different [OS](#). This is also possible for [Western Digital NAS](#) as described [here](#) and [here](#). Also the variant with a mini PC and separated storage case was interesting. The idea is to get the most out of money, so the ranking here is done by price/performance ratio and the DIY [NAS](#) wins the comparison.

- DIY [NAS](#) system on PC base
- DIY [NAS](#) system with separated storage case and mini PC
- Commercial [NAS](#) system
- Commercial [NAS](#) with custom OS

### 2.3 The planned hardware

When comparing the different hardware solutions, I also made a comparison between an Intel and an AMD based [NAS](#). The AMD variant won the price/power chapter, so I decided to buy:

- 32GB G.Skill RipJaws V black DDR4-3200 DIMM Dual Kit
- 250GB Samsung 970 Evo Plus M.2 2280
- 400 Watt be quiet! Pure Power 11 CM Modular 80+ Gold

- Black Fractal Design Node 304 cube without power supply
- 4x 4000GB WD Red Plus WD40EFZX 128MB 3.5"
- ASRock Fatal1ty B450 Gaming-ITX/AC AMD B450
- AMD Athlon 3000G with Radeon Vega Graphics 3.5GHz
- Noctua NH-L9a-AM4 topblow cooler

## 2.4 The bought hardware

„Life is what happens to you while you're busy making other plans.“

Figure 1: John Lennon

This means that the availability of chips and other events affect the market and make it difficult to realize these plans. For example, motherboards in mini-ITX format with AM4 socket are really hard to get. That's why I revised my decision on the components:

- 32GB G.Skill RipJaws V black DDR4-3200 DIMM Dual Kit
- 250GB Samsung 970 Evo Plus M.2 2280
- 400 Watt be quiet! Pure Power 11 CM Modular 80+ Gold
- Black Fractal Design Node 304 cube without power supply
- 4x 4000GB WD Red Plus WD40EFZX 128MB 3.5"
- ASRock H610M-ITX/AC mITX Intel H610 DDR4 S1700
- Intel Core i3-12100 tray
- Noctua NH-L9i-17xx topblow cooler

The difference is that the CPU is a latest generation [Intel i3](#) processor. The corresponding [motherboard](#) is also brand new. Both were only launched in Q4/2022. This has some consequences: [OMV](#) 5 does not support the network chip of the motherboard. So I decided to install the [OMV](#) 6 beta on my [DIY NAS](#). I also tried [TrueNAS](#) – it looks much more professional than [OMV](#), but it is also a bit more complicated to understand and configure. It also doesn't have native support for [Docker](#).

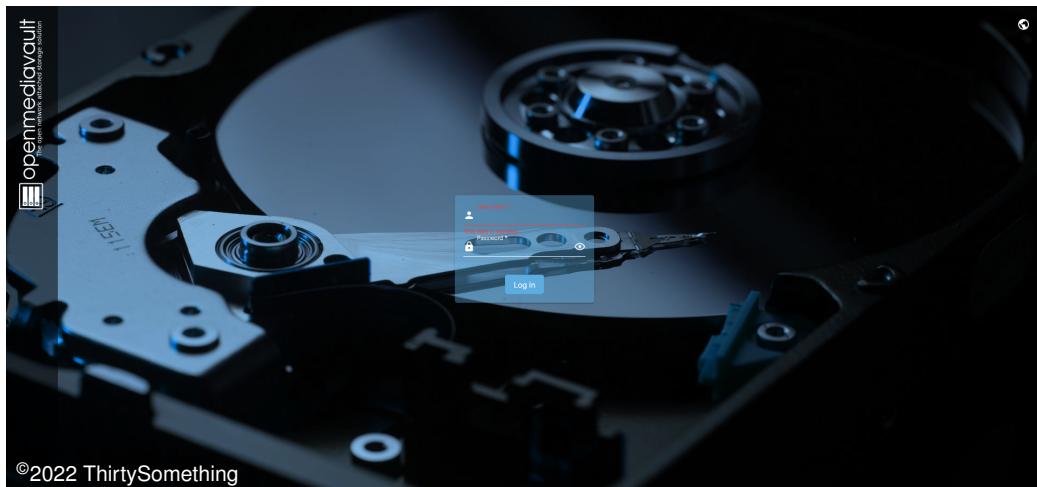
### 3 The hardware build

Although I build up my last PC more than 25 years ago assembling the hardware was a breeze. The only surprise was the mounting position of the power supply. As I understand how to do this everything was fine.

## 4 The software installation

### 4.1 The operating system

The basic installation of [OMV](#) works without any problems. This is known from debian. Additional [Volker Theile](#), the founder of the [OMV](#) project, and his team have done a very good job. Thank you guys!



**Figure 2:** The OMV login page

### 4.2 The disks

All attached disks are working. The system disk is a NVME SSD with a 250 GB capacity. All others are 4 TB WD RED disks.

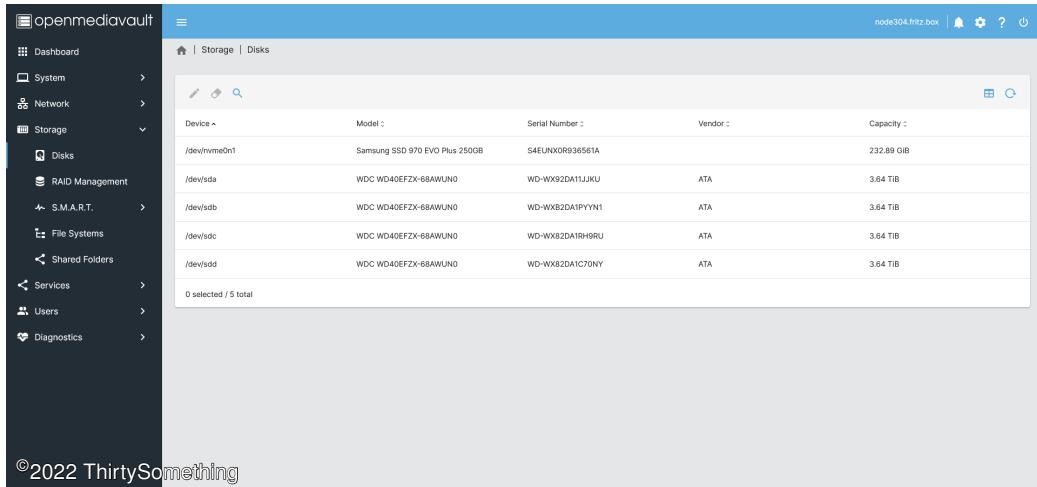
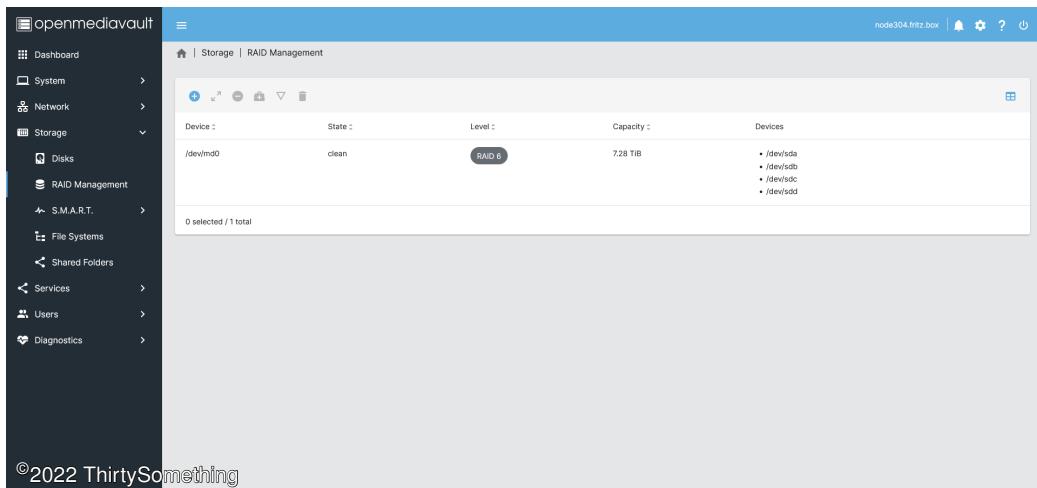


Figure 3: The OMV disks

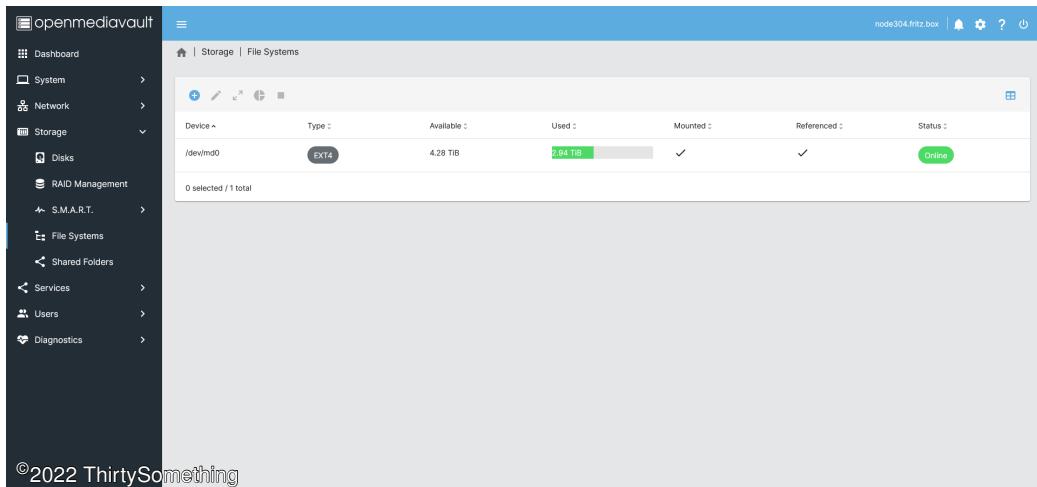
### 4.3 The RAID storage

Configuring the [RAID](#) was also less problematic. But there is an important point: As long as the RAID system is created, do not use the RAID! During the RAID build I was playing around with [OMV plugins](#) to see the diskstats – and damaged the RAID build. After a reboot the system hangs on the filesystem check. I was shocked - also about the large amount of blocks the fsck found and tried to repair. It took a while to understand what happens. Then I've startet from scratch and everything was okay. The creation of the file system at the end went without problems. A filesystem check found nothing to do.

**Figure 4:** The OMV RAID

## 4.4 The file system

This was a simple step – just create a filesystem on the previously created RAID volume.

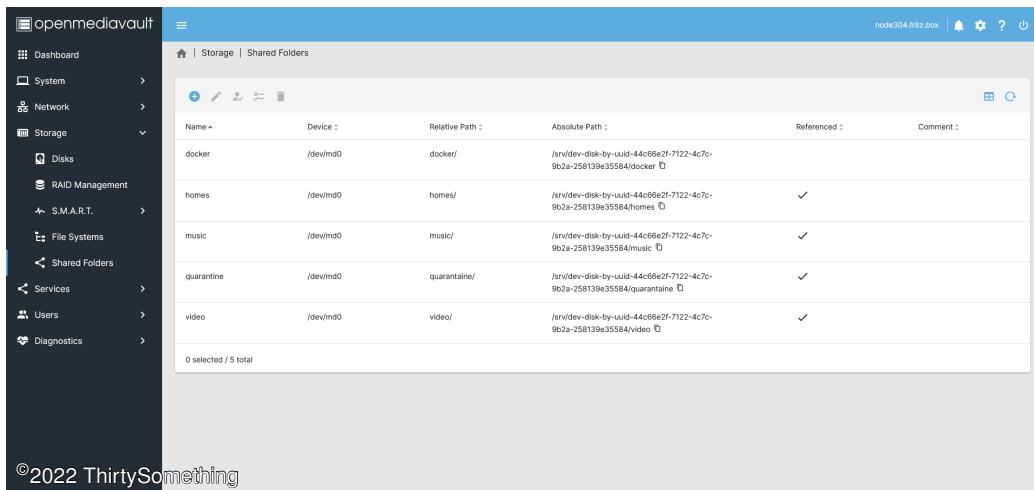
**Figure 5:** The OMV filesystem on the RAID

## 4.5 Shared folders

Some common folders should be defined before anything else is done:

- The folder `docker` for the use of [Docker](#).

- The folder `homes` as base folder for the users home directories.
- The folder `music` for the miniDLNA plugin.
- The folder `quarantine` for the use of [ClamAV](#).
- The folder `video` also for the miniDLNA plugin.

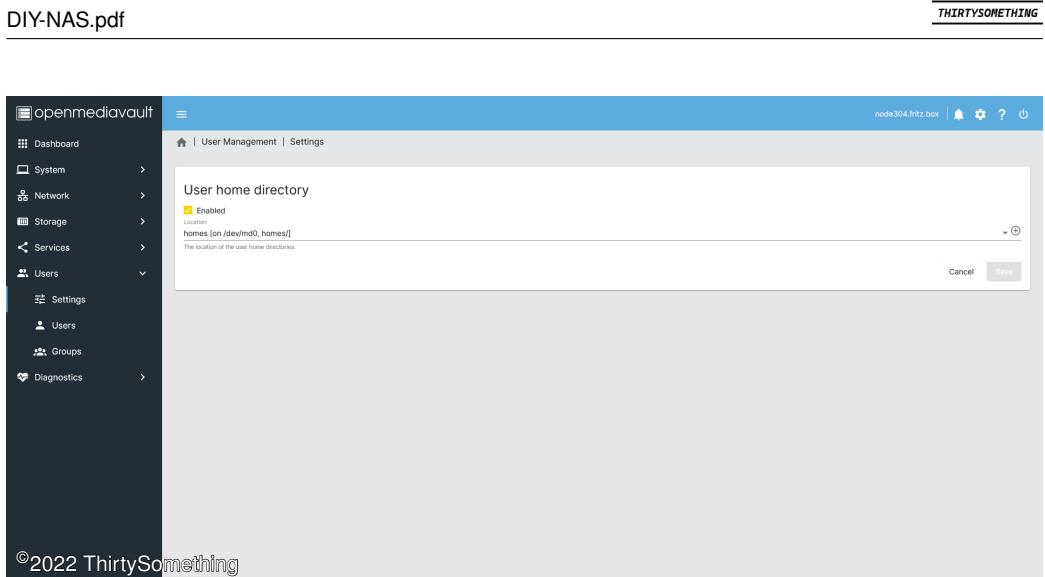


**Figure 6:** The OMV shared folders

This „shared folders“ are defined as container to be used inside of [OMV](#). To make them accessible from the network you have to enable services for them.

## 4.6 Users

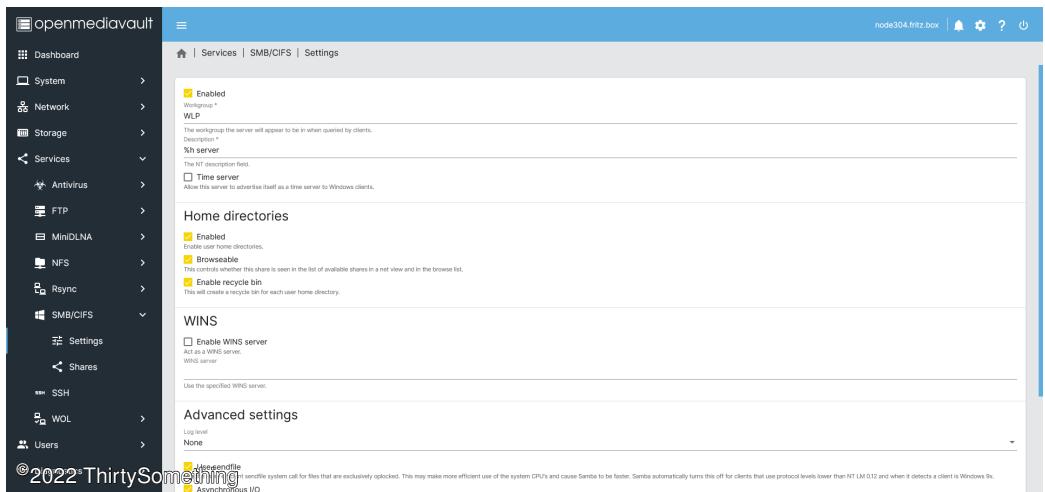
In the settings the option `User home directory` is enabled and points to the previously created `homes` folder. Then I created the users.



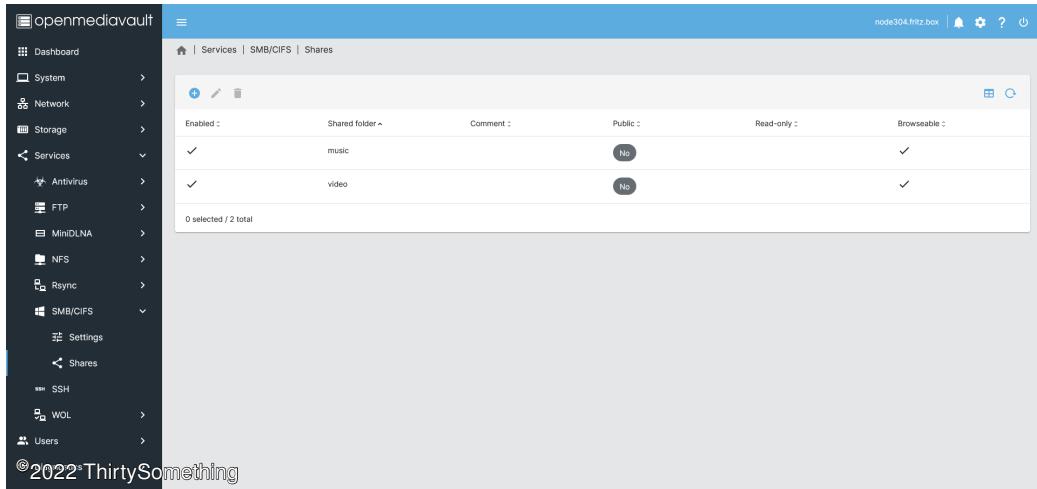
**Figure 7:** The OMV users home directory

## 4.7 CIFS shares

Simple - enable the SMB/CIFS service, enable the home directories and then create shares for the previously defined shared folders. Allow read/write access for the administrator of the shares (me), the others got read access to them.



**Figure 8:** The OMV CIFS settings

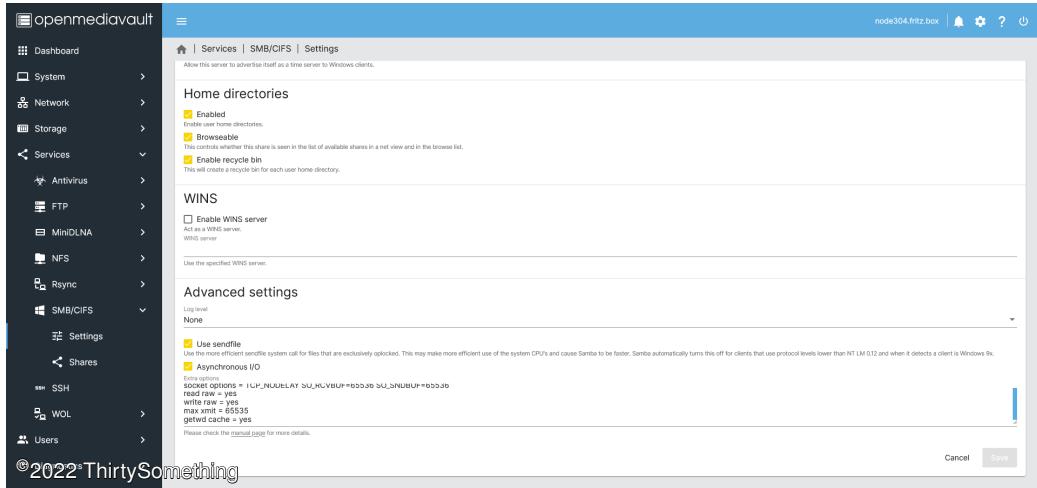
**Figure 9:** The OMV CIFS shares

As you can see there is actually no public access to these shares. This means that only privileged users can access them.

## 4.8 Tuning

### 4.8.1 SMB/CIFS

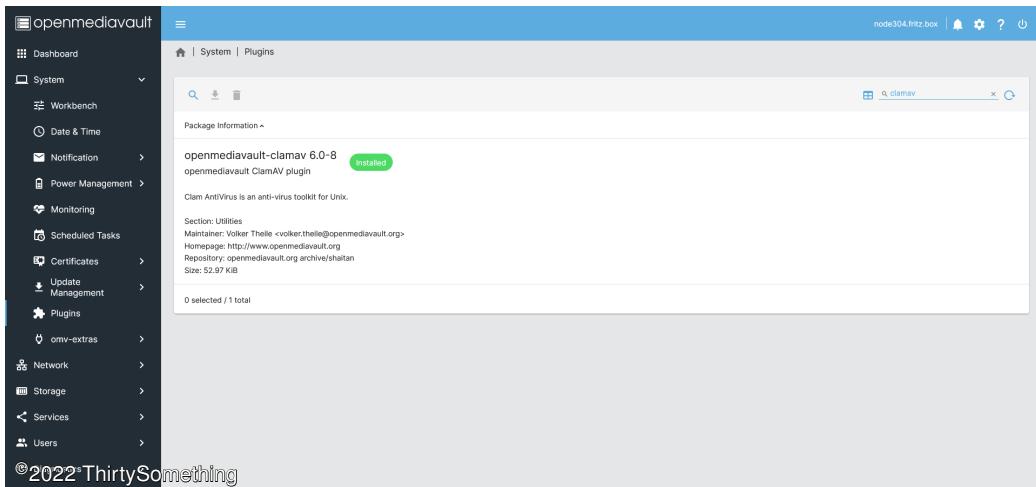
Transferring data from the old **NAS** to the new one takes a lot of time. To speed up the process I searched for some tuning and found this [here](#):

**Figure 10:** The OMV advanced CIFS settings

## 5 OMV plugins

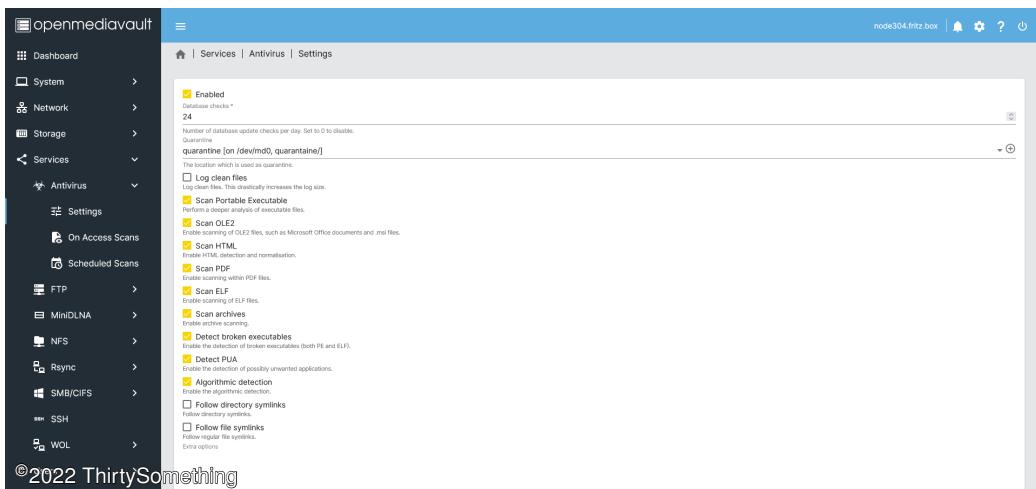
### 5.1 ClamAV

To protect the data I want to use an antivirus program. As open source solution there is [ClamAV](#) available – and also as plugin for [OMV](#).



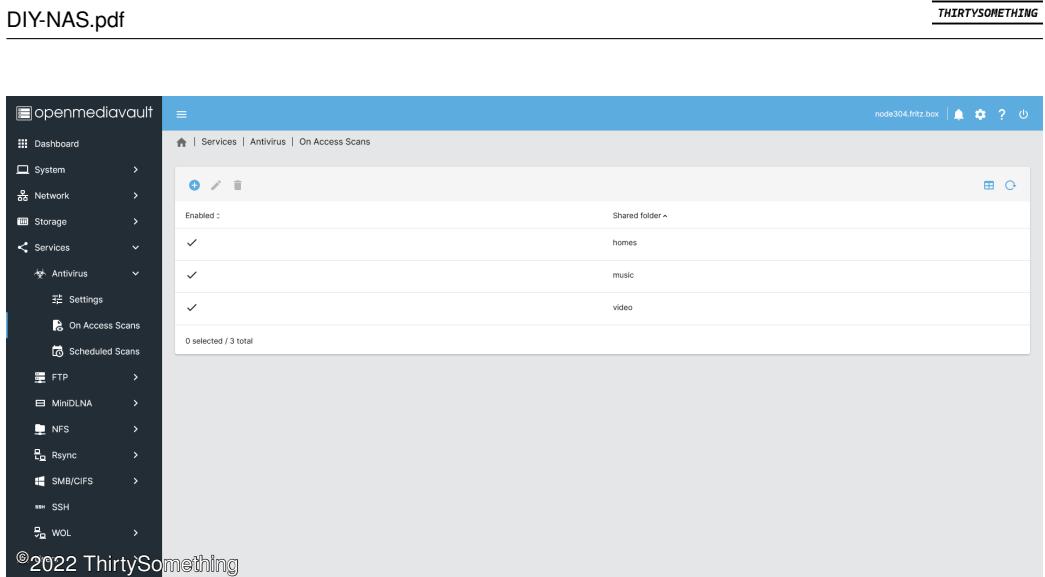
**Figure 11:** The OMV ClamAV plugin

In the setup we use the previously defined quarantine folder.



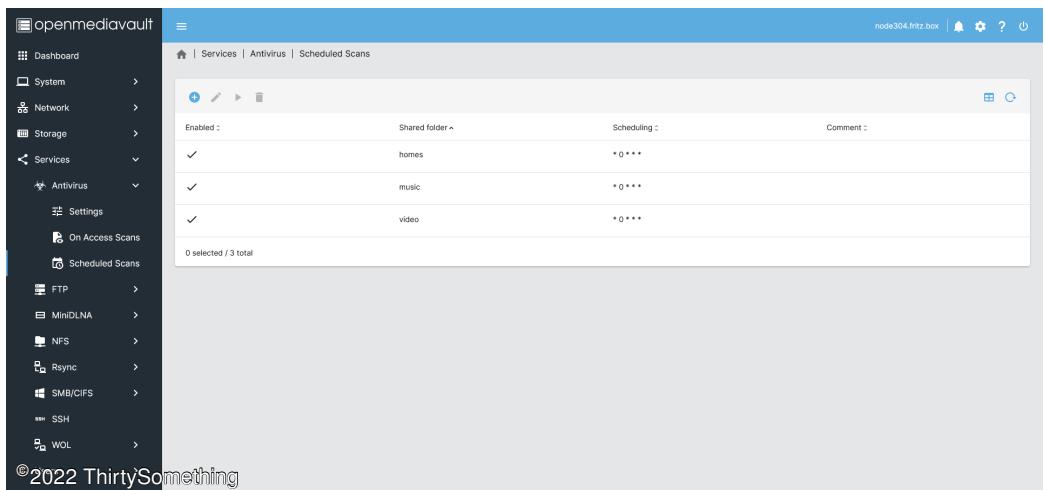
**Figure 12:** The antivirus settings

I enabled a scan on access for specific folders.



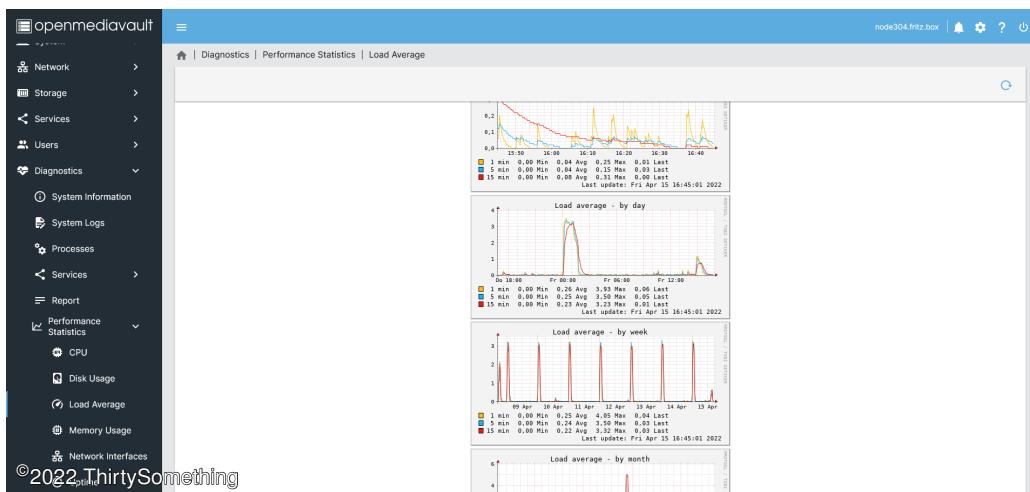
**Figure 13:** The antivirus on access scan

Also I've enabled a scheduled scan for these folders, too.



**Figure 14:** The antivirus scheduled scan

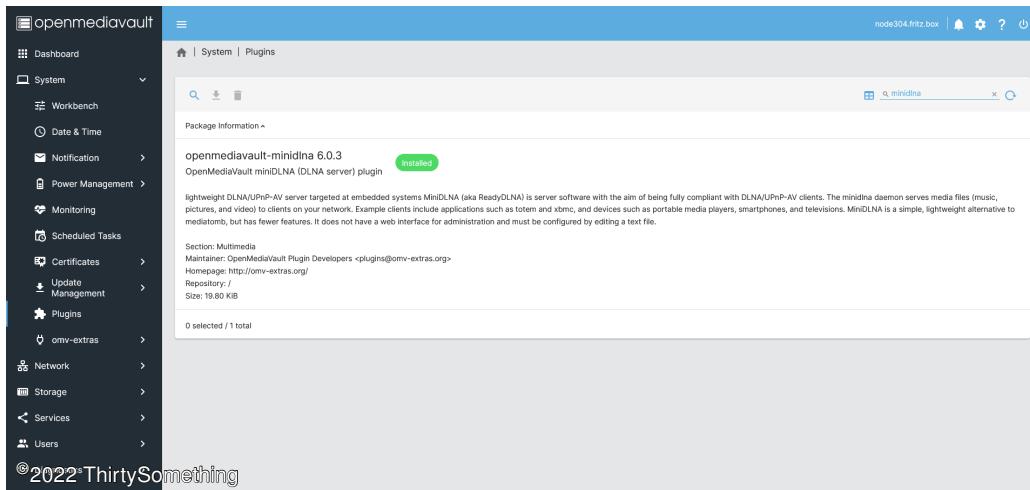
Maybe somebody will claim that all scheduled scans run on the same time. Yes – this was setup to check the power of the CPU. The system load increases to a load of about 3 – that's fantastic from my point of view! This means that there are more than enough reserves. When I'm spreading the schedule I can lower the load.



**Figure 15:** The antivirus system load on scan

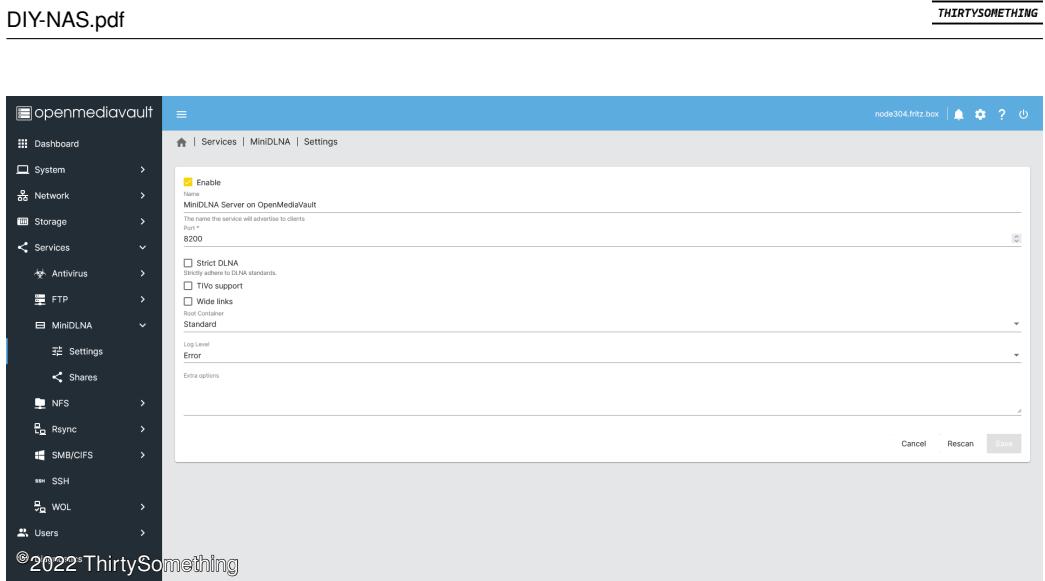
## 5.2 MiniDLNA

To stream music and videos from the [NAS](#) to the network I'm using the [ReadyDLNA](#) media server software. In [OMV](#) this is used with the old name [Minidlna](#).



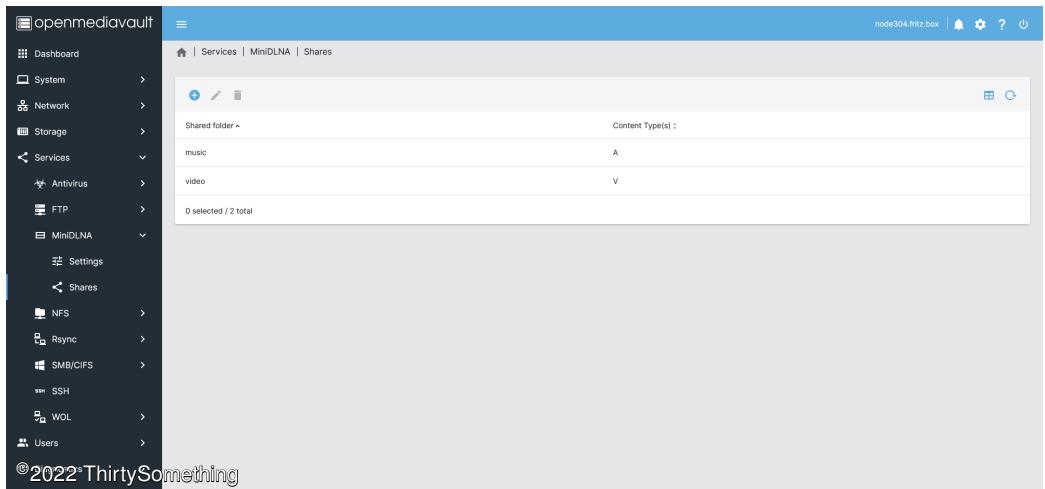
**Figure 16:** The MiniDLNA plugin

The basic setup is simple - I've checked the Enable box and that's it.



**Figure 17:** The MiniDLNA settings

Then I have to define the shares and the kind of content of the shares.



**Figure 18:** The MiniDLNA shares

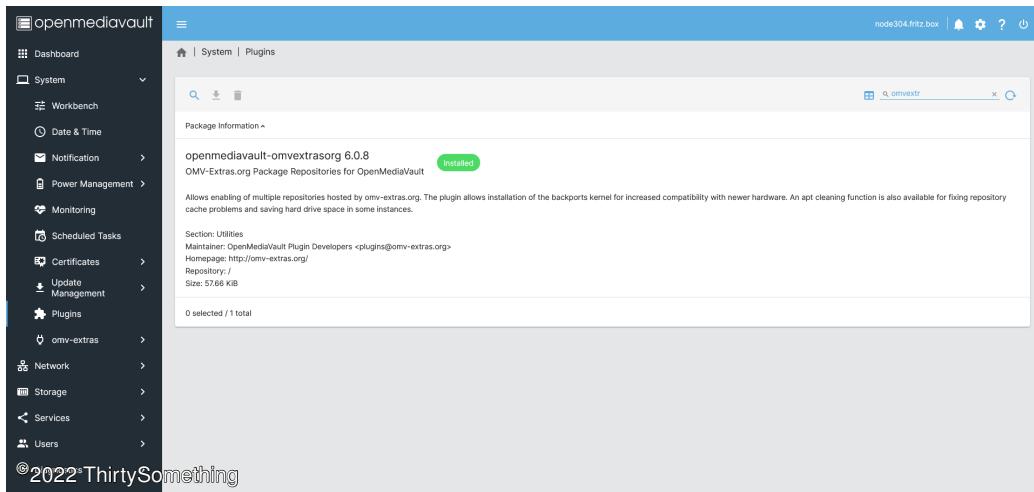
### 5.3 OMV extras

The **OMV extras** are not available in the plugin list. The way to go to install them is described [here](#). Login as user `root` using SSH and enter the following command:

```
sudo su -
wget -O https://github.com/OpenMediaVault-Plugin-Developers/packages/raw/master/install | bash
```

**Figure 19:** The OMV extras installation

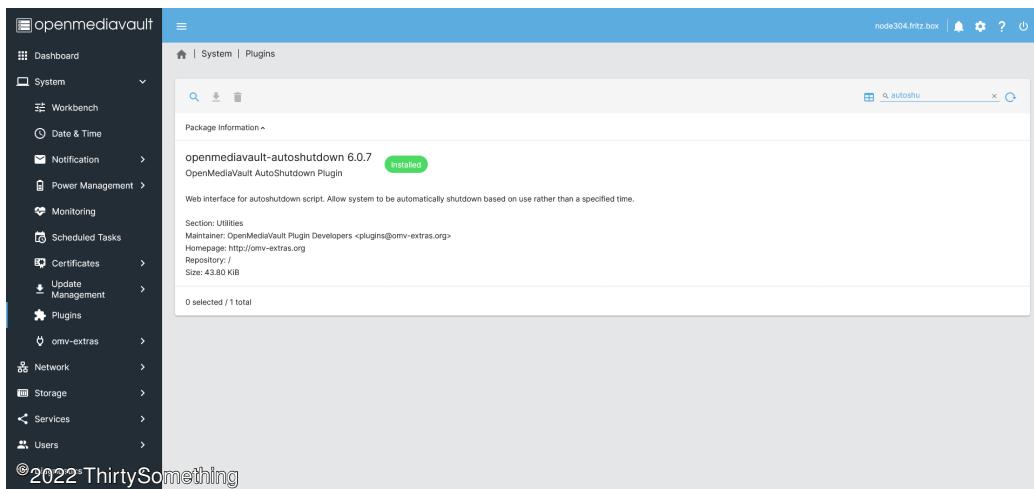
Using this plugin enables the [Docker](#) and [Portainer](#) installation to enhance the capabilities of the [NAS](#).



**Figure 20:** The OMV extras plugin

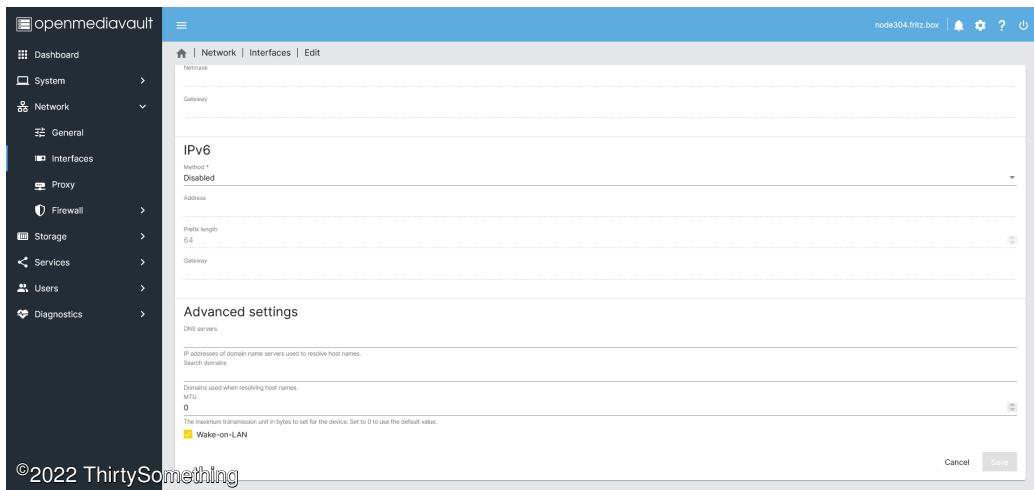
## 5.4 Autoshutdown

To save some energy we want the system to shutdown when not used and to wake up when used. So we want to use the autoshutdown plugin and the [WOL](#) feature. **NOTE:** Don't forget to enable [WOL](#) in the BIOS. For my system there have been two settings, one setting on the adapter itself and one setting in the boot options.



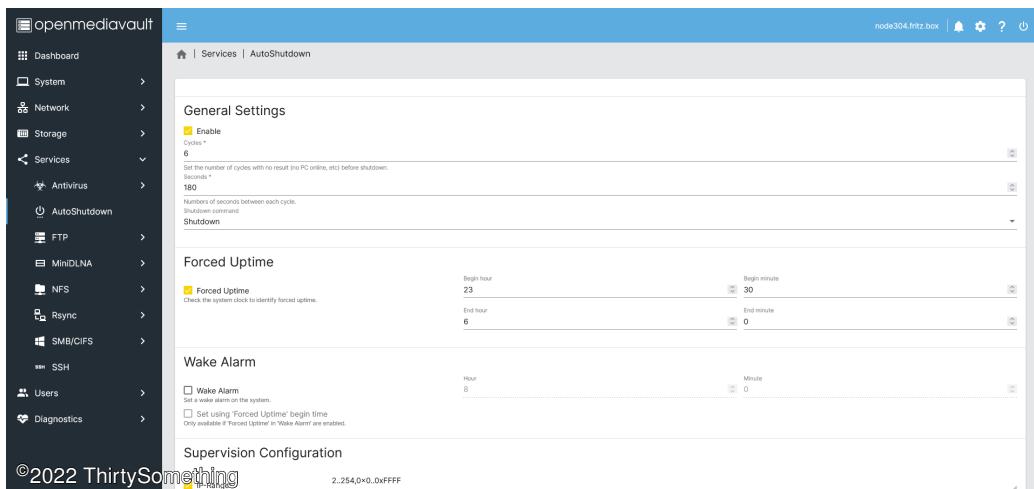
**Figure 21:** The OMV autoshutdown plugin

As first step we have to enable the **WOL** on the network card. You can find the checkbox as last option of the Advanced settings section.



**Figure 22:** The OMV network card setup

Then we have to configure/setup the autoshutdown plugin. First step is to enable the autoshutdown. Second step is to force an uptime between 23:30 and 02:00 to ensure backup is done. The rest is left with default settings and might be updated at a later point.



**Figure 23:** The autoshutdown setup 1

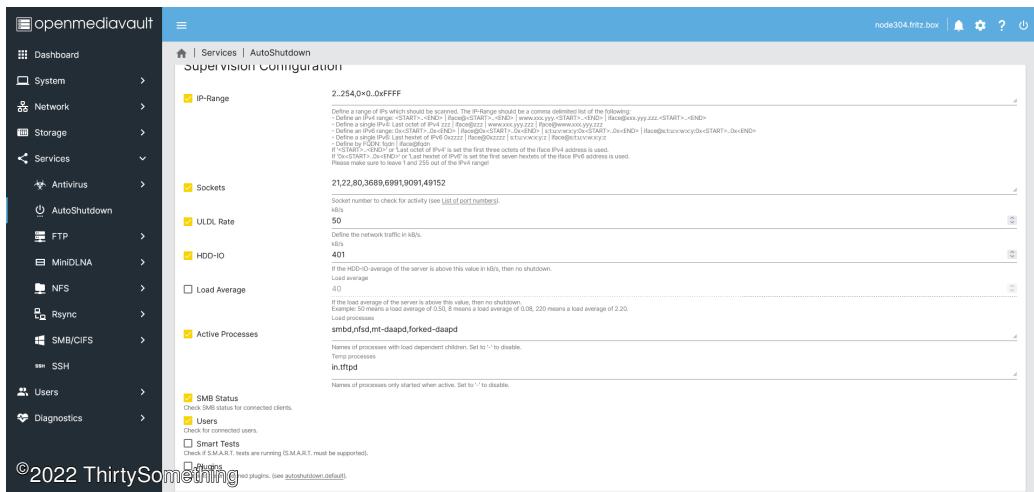


Figure 24: The autoshutdown setup 2

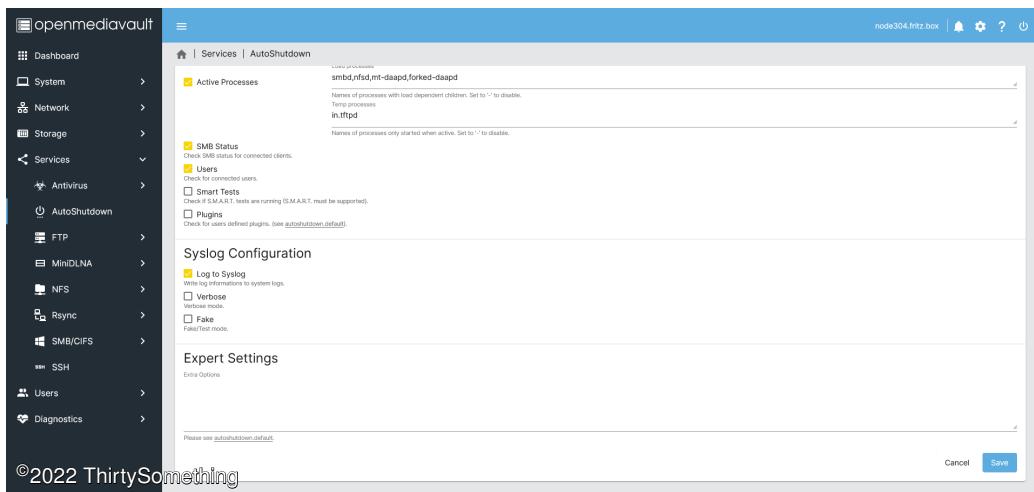
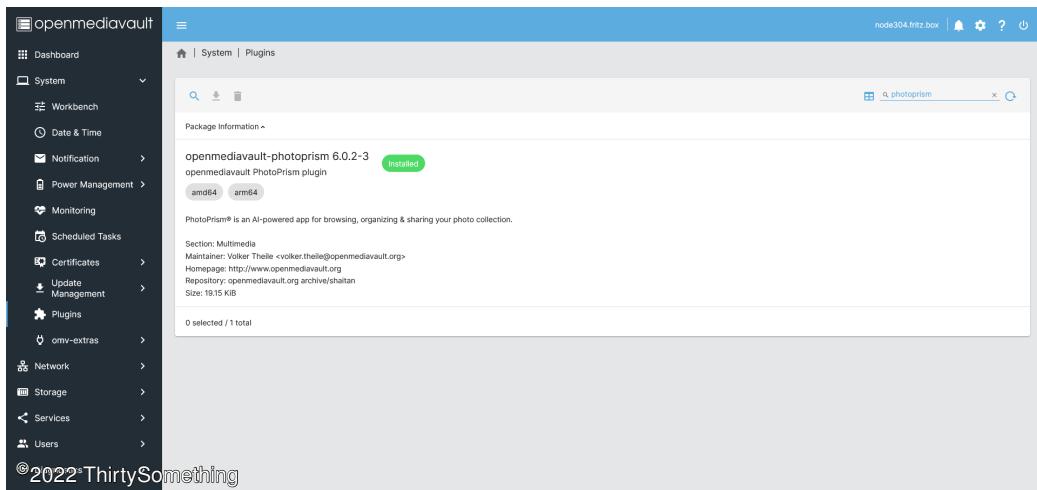


Figure 25: The autoshutdown setup 3

## 5.5 PhotoPrism

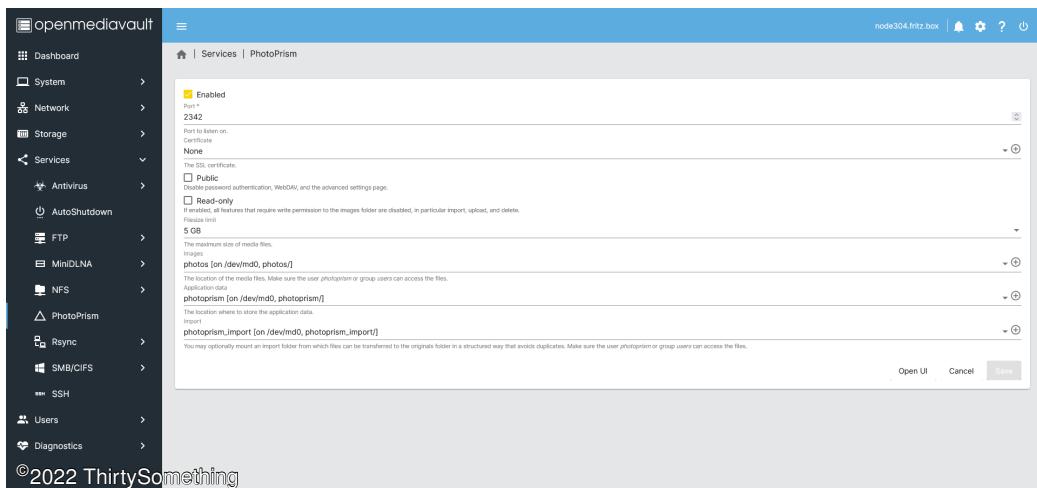
I used the app [PhotoPrism](#) to de-duplicate my images. The last 15 years a lot of duplicates blowed up the collection. Let's start with the first step, the installation of the plugin.



**Figure 26:** The OMV PhotoPrism plugin

The next step has been the creation of the following shared folders:

- **photoprism** – The location for the image database.
- **photoprism\_import** – The share used for image import.
- **photos** – The final location of the photos in an organized way.



**Figure 27:** The OMV PhotoPrism setup

The setup is easy, just set the shares – that's all folks. Okay – I remember some images I made with a digital SLR camera. They are large as I remember, so I've increased the filesize limit to 5 GB.



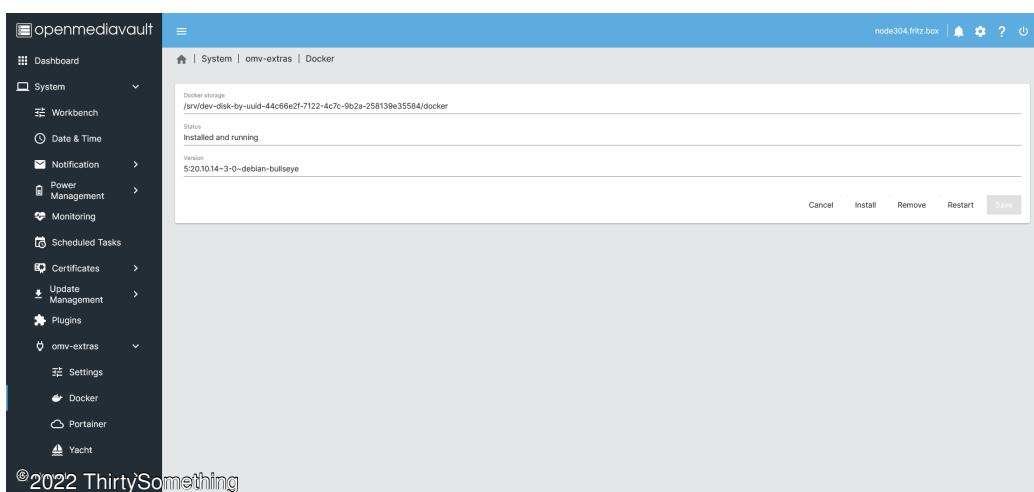
**Figure 28:** The PhotoPrism login

## 6 Docker

By using [Docker](#) I want to enhance the [NAS](#) with features which are not available out-of-the-box. Especially with services my previous [NAS](#) offers and which are not native available for [OMV](#).

### 6.1 Docker installation

The default location of [Docker](#) is `/var/lib/docker` on the system disk. Although the system disk is a fast SSD, I want to install [Docker](#) on the slower RAID storage. So I have to change the path of [Docker](#) storage.

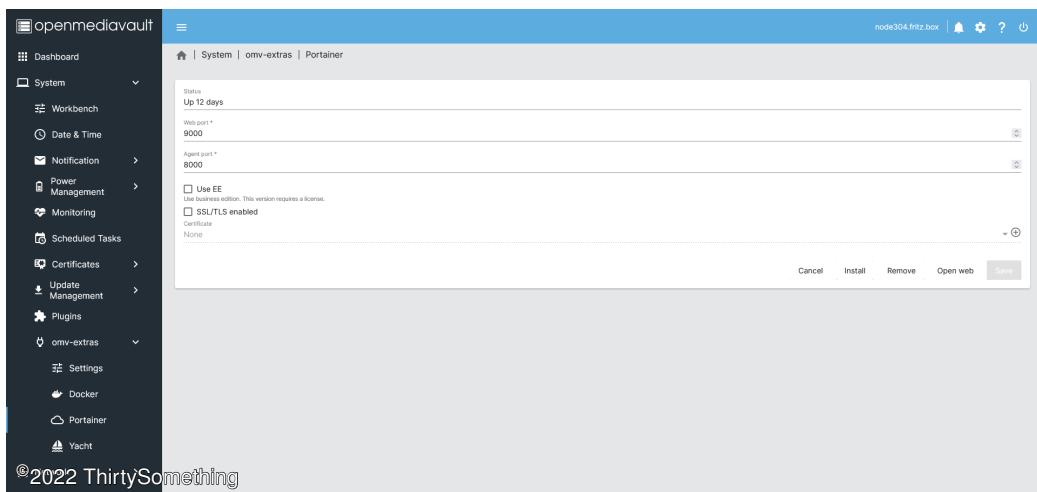


**Figure 29:** The Docker setup

In case [Docker](#) is already installed, see [here](#) how to move the [Docker](#) storage to another location.

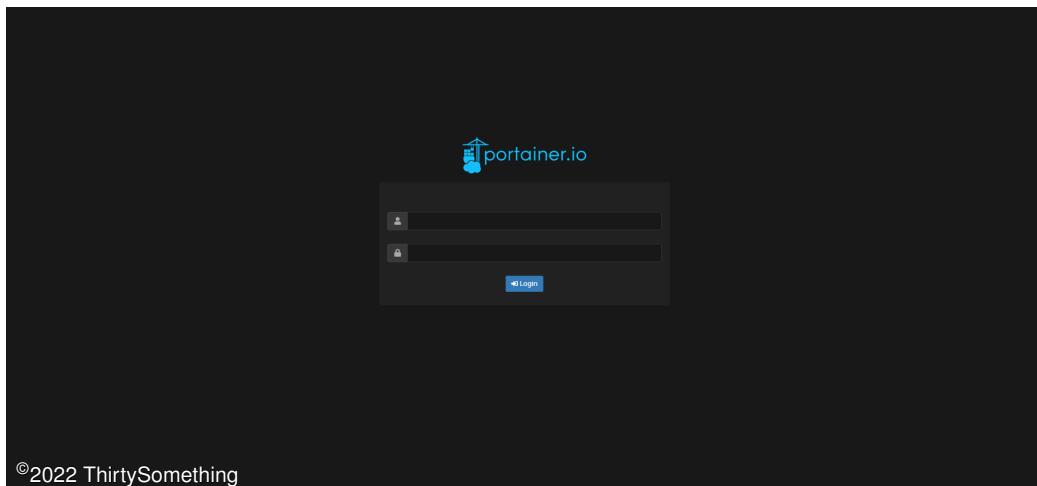
### 6.2 Portainer

To have more comfort in dealing with [Docker](#) we install also [Portainer](#) from the [OMV extras](#).



**Figure 30:** The Portainer setup

After installation [Portainer](#) is up and running.



**Figure 31:** The Portainer login

### 6.3 Container list

- [Portainer](#) — UI for docker – done with docker installation.
- [SCM-Manager](#) — SCM with git, svn and hg – done, works fine.
- [MariaDB](#) — Database for various projects – done, works fine.
- [phpMyAdmin](#) — UI for the database – done, works fine.
- [Mosquitto](#) — A MQTT broker
- [Syncthing](#) — File synchronization – done.
- [Nextcloud](#) — Own cloud service
- Backup server?
  - [restic](#)
  - [Duplicati](#)
- [MinIO](#) — As data sink for backup?
- [Pi-hole](#) — Adblocker for the network – won't do that, see section [6.4](#) for explanation
- [gotify](#) — Notification server

**Figure 32:** Containerlist

### 6.4 Pi-hole

I'm already using [Pi-hole](#) on a old [Raspberry Pi 2B](#) and I'm very pleased with that software. Because of it's nature working as a [DNS resolver](#) I won't move this to the NAS. The reason is quite simple – this will influence the [Autoshutdown](#) plugin and/or will make it obsolete. This is not what I want to achieve.

### 6.5 SCM-Manager

[SCM-Manager](#) provides a comfortable user interface for [git](#), [Mercurial](#) and [Subversion](#). Up to now I use [Subversion](#) for [version control](#). Some of my

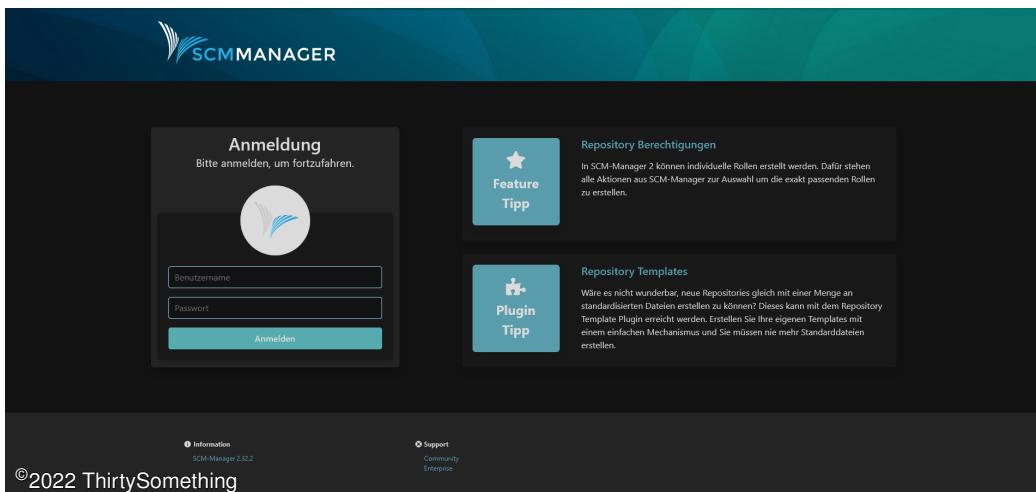
repositories are private and I will never publish them to a public hoster like [GitHub](#) although they offer private repositories. All my other repositories are hosted on my [NAS](#) – the plan is to move them to [git](#) if possible – just to have them on a more modern version control system.

The definition for [Docker compose](#) looks like:

```
version: '3.7'
services:
  scmmanager:
    image: clodogu/scm-manager:latest
    restart: unless-stopped
    ports:
      - 2222:2222
      - 8080:8080
    volumes:
      - scm-config:/var/cache/scm/work
      - scm-data:/var/lib/scm
  volumes:
    scm-config:
    scm-data:
```

**Figure 33:** docker-compose.yaml for SCM-Manager

The startup of the container should work without any problems.



**Figure 34:** The SCM-Manager login

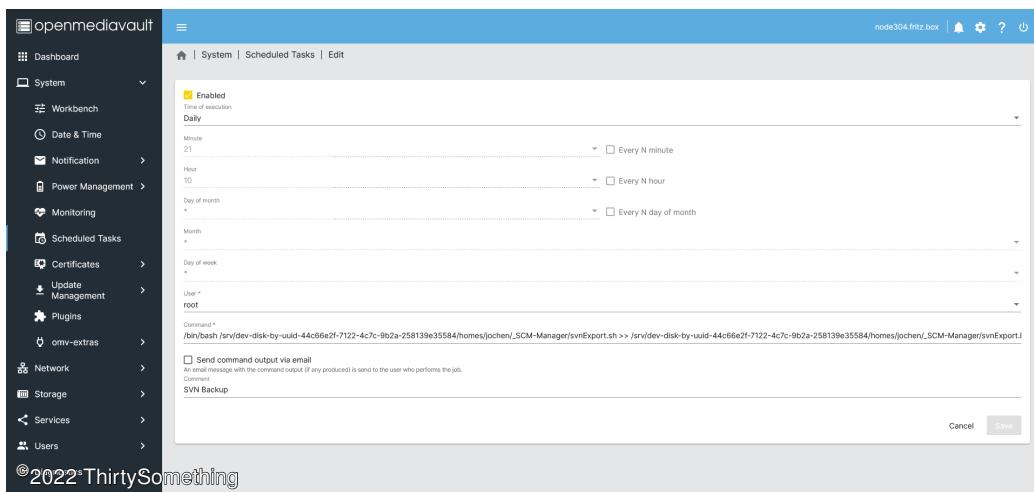
After importing my repositories, I've setup a backup job. For this I've updated my script [svnExport.sh](#) that it will work for [SCM-Manager](#). Setting up a cronjob inside a [Docker](#) container is a [pita](#). The previously created volume is required to dynamically export all existing [Subversion](#) repositories. The command will be something like this one:

```
/bin/bash /srv/<omv-raid>/<path-to-script>/svnExport.sh >>
/srv/<omv-raid>/<path-to-script>/svnExport.log 2>&1 &
```

**Figure 35:** Cron command for svnExport

For the script please see appendix [A](#).

The job is then setup in [OMV](#) scheduled tasks accessing the script. **NOTE:** The path to the [SCM-Manager](#) repositories needs to be set inside in the script.



**Figure 36:** The SCM-Manager backup task

## 6.6 Syncthing

[Syncthing](#) is a file synchronization service. I use this to backup automatically data from my smartphone to the PC and then from the PC to the NAS. The setup is easy regarding the [Docker README](#) of the project.

First of all I've created a [Docker](#) volume. In the next step I've setup the container regarding the ports and map the created volume to `/var/syncthing`. An important step is to set the environment variables `PUID` and `GUID`. Then the container will run unless stopped.

Setting up the synchronized folders is as usual with [Syncthing](#). The result may look like this one.

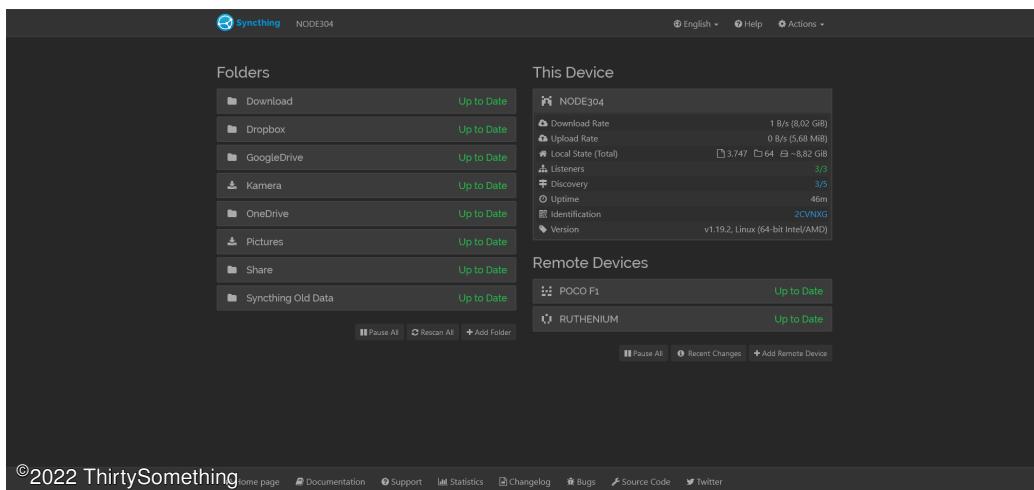


Figure 37: Syncthing

## 6.7 Watchtower

The [Watchtower](#) container is very simple to install. There is no need for a volume. And – they offer on their website a `docker-compose.yaml` file. Just copy this definition to a new Portainer stack, deploy and run. I've extended the default file with the settings of email notifications.

```
version: "3"
services:
  watchtower:
    image: containrrr/watchtower
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      # WATCHTOWER_MONITOR_ONLY: 'true'
      WATCHTOWER_NOTIFICATIONS: email
      WATCHTOWER_NOTIFICATION_EMAIL_FROM: developer@derpaul.net
      WATCHTOWER_NOTIFICATION_EMAIL_TO: developer@derpaul.net
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER: <smtp server>
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PORT: 587
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER_USER: developer@derpaul.net
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PASSWORD: <top secret password>
      WATCHTOWER_NOTIFICATION_EMAIL_DELAY: 2
    restart: unless-stopped
```

Figure 38: docker-compose.yaml for Watchtower

This container does not have any user interface. The purpose of this container is to keep all other containers with the `:latest` suffix up to date. In concrete means this to check for newer versions of containers and will update/restart them automagically if required. For german users there is an [introduction](#) video available.

## 6.8 MariaDB und phpMyAdmin

This will become a little bit more complicated. Basically you need three components to run all:

- The database [MariaDB](#)
- A webserver like [Apache](#), [nginx](#) or something else
- And [phpMyAdmin](#)

I chose not to install the web server separately and instead chose a Docker container that contains the web server in addition to phpMyAdmin. But let's start with [MariaDB](#) at first.

```
version: '3.7'
services:
  mariadb:
    image: mariadb:latest
    restart: unless-stopped
    environment:
      MARIADB_ROOT_PASSWORD: <top secret password here>
    volumes:
      - mariadb-data:/var/lib/mysql
    ports:
      - 3306:3306
volumes:
  mariadb-data:
```

**Figure 39:** docker-compose.yaml for MariaDB

Then we setup the container for [phpMyAdmin](#):

```
version: '3.1'
services:
  phpmyadmin:
    image: phpmyadmin:latest
    restart: unless-stopped
    ports:
      - 8088:80
    environment:
      PMA_HOST: <IP of your host>
```

**Figure 40:** docker-compose.yaml for phpMyAdmin

And here we go:



©2022 ThirtySomething

**Figure 41:** phpMyAdmin

## 7 Ports

It's always a good idea to remember the used ports. This section will give you an overview about the used ports and the name of the program using this port.

Program	Port	Comment
MariaDB	3306	Database connection port
OMV	80	UI
PhotoPrism	2342	UI
phpMyAdmin	8088	UI
Portainer	9000	UI
SCM-Manager	8080	UI
SCM-Manager	2222	ssh
Syncthing	8384	UI
Syncthing	TCP:22000	Filetransfer
Syncthing	UDP:22000	Filetransfer
Syncthing	UPD:21027	Receive local discovery broadcasts

**Table 2:** Used ports - order by program

Program	Port	Comment
OMV	80	UI
SCM-Manager	2222	ssh
PhotoPrism	2342	UI
MariaDB	3306	Database connection port
SCM-Manager	8080	UI
phpMyAdmin	8088	UI
Portainer	9000	UI
Syncthing	8384	UI

Syncthing	UPD:21027	Receive local discovery broadcasts
Syncthing	TCP:22000	Filetransfer
Syncthing	UDP:22000	Filetransfer

**Table 3:** Used ports - order by port

## A svnExport.sh

```
#!/bin/bash
#+-----+
#| Script to export configured svn repositories
#+-----+
# Comment line out for debugging purposes
# set -x
#+-----+
#| Variable definitions
#+-----+
# location of all repositories
VAR_PATH SVN="/srv/dev-disk-by-uuid-44c66e2f-7122-4c7c-9b2a-258139e35584/docker/volumes/scmmanager/_data/repositories/*"
# suffix for SCM organized repositories
SUFFIX SVN=data
# filename containing real repository name
META SVN=metadata.xml
# maximum days to keep a backup
INT_AGE=5
# file extension
STR_EXT=gz
# current date
STR_DATE=$(date +%Y-%m-%d)
# get current name of backup folder
DIR_EXPORT=$(dirname "${0}")
DIR_EXPORT=$(realpath "${DIR_EXPORT}")

#+-----+
#| Check for valid SVN repository
#+-----+
function is_svn_repository {
    svnlk info "${1}" >/dev/null 2>&1
    echo $?
}

#+-----+
#| Get base name for SVN repository
#+-----+
function get_svn_base_name {
    REPOBASE=$(dirname "${1}")/${META SVN}
    REPONAME=$(xmlstarlet sel -T -t -m "/repositories/name" -v "/repositories/name" < "${REPOBASE}")
    echo "${REPONAME}"
}

#+-----+
#| Create name for SVN repository for export
#+-----+
function get_svn_destination_name {
    REPOBASE=$(get_svn_base_name "${1}")
    echo "${DIR_EXPORT}/${REPOBASE}-${STR_DATE}.${STR_EXT}"
}

#+-----+
#| Delete backups older than specified age
#+-----+
function drop_old_exports {
    PATTERN=$(get_svn_base_name "${1}")
    FTK=$((1 + ${INT_AGE}))
    echo "$(date +'%Y%m%d-%H:%M:%S'): Keep the last ${INT_AGE} backups"
    COUNTER=0
    for CURRENT_DUMP in $(find "${DIR_EXPORT}" -name "${PATTERN}.*.${STR_EXT}" | sort -nr); do
        if [[ "${CURRENT_DUMP}" -lt "${FTK}" ]]; then
            echo "Keep dump [${CURRENT_DUMP}]"
        else
            echo "Delete dump [${CURRENT_DUMP}]"
            rm "${CURRENT_DUMP}"
        fi
        COUNTER=$((COUNTER + 1))
    done
    echo ""
}

#+-----+
#| Export SVN repository
#+-----+
function export_svn_repository {
    VAR_DEST_NAME=$(get_svn_destination_name "${1}")
    echo "$(date +'%Y%m%d-%H:%M:%S'): Dumping repo [${1}] to [${VAR_DEST_NAME}]"
    svnadmin dump "${1}" | gzip > "${VAR_DEST_NAME}"
}

#+-----+
```

```
#| Loop over all repositories
#+-----
echo "-***** Start export of SVN repositories for date [${STR_DATE}]"
for VAR_CURRENT_DIR in ${VAR_PATH SVN}; do
    SVN_REPO="${VAR_CURRENT_DIR}/${SUFFIX SVN}"
    if [[ ${is_svn_repository "${SVN_REPO}"} -eq 1 ]]; then
        echo "${(date +'%Y%m%d-%H:%M:%S')}: Skip non SVN repository [${SVN_REPO}]."
        echo ""
    else
        export_svn_repository "${SVN_REPO}"
        drop_old_exports "${SVN_REPO}"
    fi
done
echo ""
```

## List of Figures

1	John Lennon	6
2	The OMV login page	8
3	The OMV disks	9
4	The OMV RAID	10
5	The OMV filesystem on the RAID	10
6	The OMV shared folders	11
7	The OMV users home directory	12
8	The OMV CIFS settings	12
9	The OMV CIFS shares	13
10	The OMV advanced CIFS settings	13
11	The OMV ClamAV plugin	14
12	The antivirus settings	14
13	The antivirus on access scan	15
14	The antivirus scheduled scan	15
15	The antivirus system load on scan	16
16	The MiniDLNA plugin	16
17	The MiniDLNA settings	17
18	The MiniDLNA shares	17
19	The OMV extras installation	17
20	The OMV extras plugin	18
21	The OMV autoshutdown plugin	18
22	The OMV network card setup	19
23	The autoshutdown setup 1	19
24	The autoshutdown setup 2	20
25	The autoshutdown setup 3	20
26	The OMV PhotoPrism plugin	21
27	The OMV PhotoPrism setup	21
28	The PhotoPrism login	22
29	The Docker setup	23
30	The Portainer setup	24
31	The Portainer login	24
32	Containerlist	25
33	docker-compose.yaml for SCM-Manager	26
34	The SCM-Manager login	26
35	Cron command for svnExport	27
36	The SCM-Manager backup task	27
37	Syncthing	28
38	docker-compose.yaml for Watchtower	28
39	docker-compose.yaml for MariaDB	29

40	docker-compose.yaml for phpMyAdmin . . . . .	29
41	phpMyAdmin . . . . .	30

## List of Tables

1	Change history . . . . .	3
2	Used ports - order by program . . . . .	31
3	Used ports - order by port . . . . .	32

## Glossary

**ClamAV** ClamAV, a open source antivirus software 13

**Docker** Container based virtualization 4, 5, 9, 17, 22, 25, 26

**Docker compose** Definition of container using Docker compose 25

**git** git, a version control system 24, 25

**MariaDB** MariaDB, a RDBMS 28, 30

**NAS** Network Attached Storage 3–5, 12, 15, 17, 22, 25

**OMV** Open Media Vault, a NAS operating system 5, 7, 10, 13, 15, 22, 26, 30

**OMV extras** Open Media Vault Extras, value added plugins to OMV 16, 22

**PhotoPrism** PhotoPrism, AI powered photo manager 19, 30

**phpMyAdmin** phpMyAdmin, an UI for MariaDB 28, 30

**Pi-hole** Pi-hole, network-wide ad blocking 24

**Portainer** Portainer, a container management tool 17, 22, 23, 30

**SCM-Manager** SCM-Manager, a version control system server 24–26, 30

**Subversion** Subversion, a version control system 24, 25

**Syncthing** Syncthing, file synchronization 26, 30, 31

**Synology** Manufacturer of NAS systems 3

**WOL** WOL, starts computer on specific network signal 17, 18