V1.0.10

THIRTYSOMETHING

DIY-NAS

About my DIY-NAS

ThirtySomething

26.05.2022

Contents

Change history

Version	Date	Description	Name
1.0.0	02.04.2022	• Start with description	ThirtySomething
1.0.1	16.04.2022	• Rename section WOL to Autoshutdown • Fill section Autoshutdown with content	ThirtySomething
1.0.2	16.04.2022	• Split file DIY-NAS-Content into separate files	ThirtySomething
1.0.3	21.04.2022	• Update to latest template version • Add section about used ports • Add svnExport.sh as appendix	ThirtySomething
1.0.4	23.04.2022	• Add documentation about Syncting docker container	ThirtySomething
1.0.5	25.04.2022	• Split section OMV plugins into files • Add description of PhotoPrism	ThirtySomething
1.0.6	28.04.2022	• Add list of containers from old document	ThirtySomething
1.0.7	30.04.2022	• Use latest template version • Add section about Pi-hole	ThirtySomething
1.0.8	11.05.2022	• Use latest template version • Add copyright hint to the images	ThirtySomething
1.0.9	19.05.2022	• Add description of Watchtower	ThirtySomething
1.0.10	21.05.2022	• Add description of MariaDB and phpMyAdmin • Switch SCM-Manager from manual to docker-compose.yaml • Switch syncthing from manual to docker-compose.yaml • Add description of Mosquitto	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: A quote from 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 Q4/2021. 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!

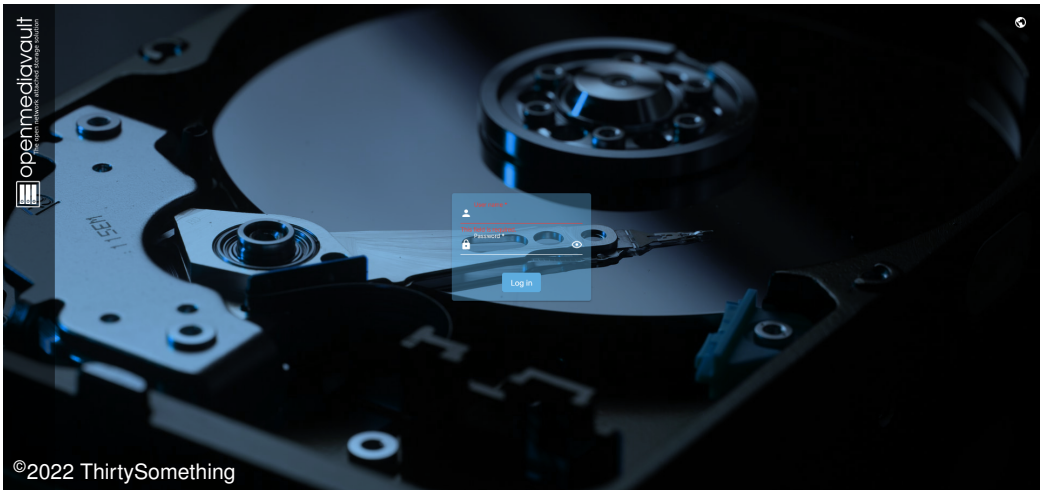

Figure 2: The OMV login page

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.

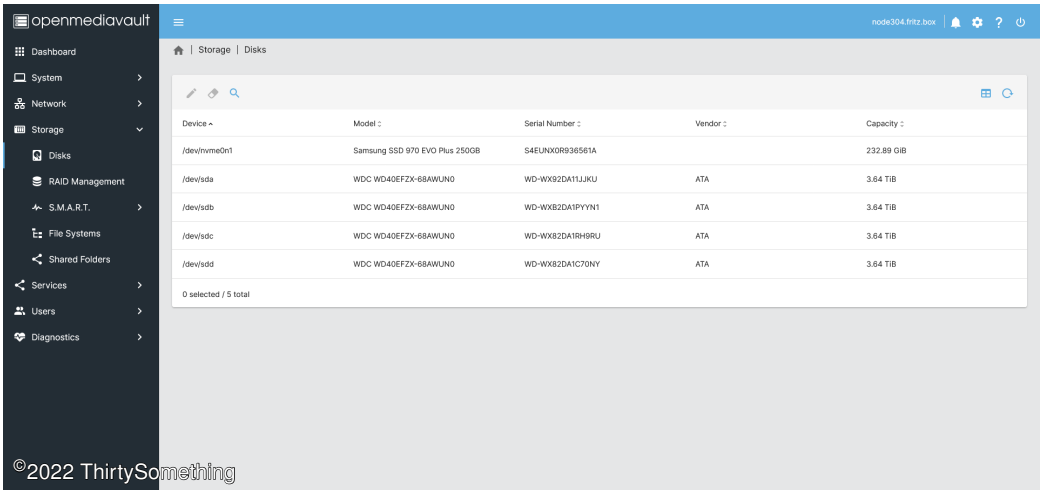

Figure 3: The OMV disks

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.

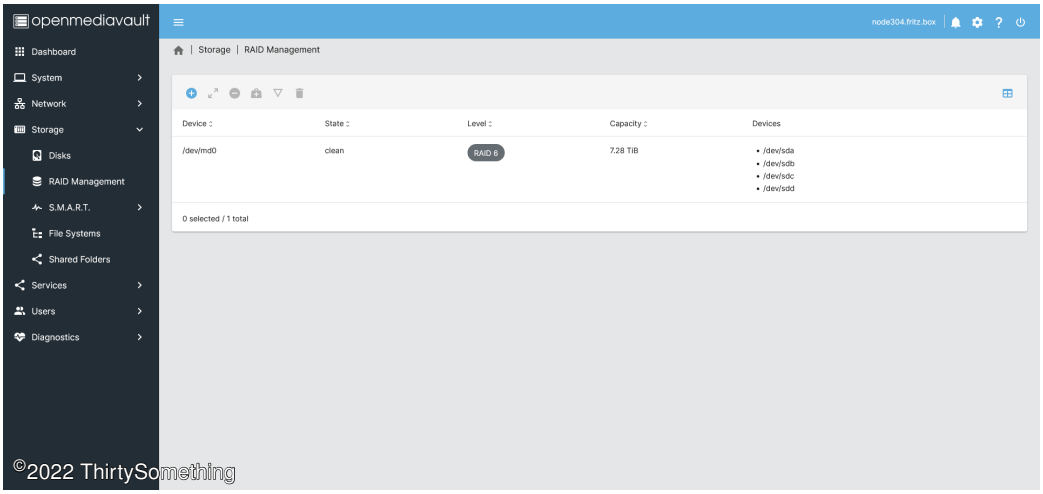

Figure 4: The OMV RAID

4.4 The file system

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

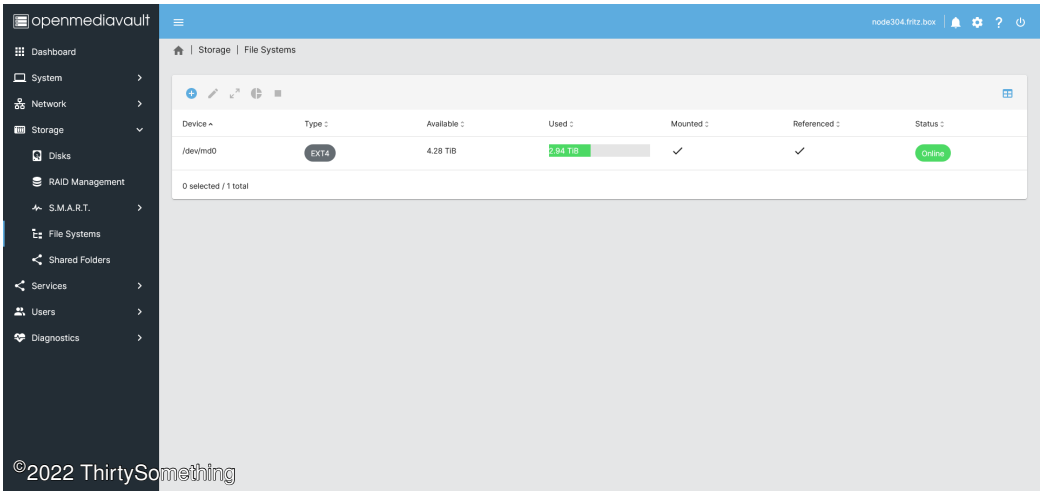

Figure 5: The OMV filesystem on the RAID

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.

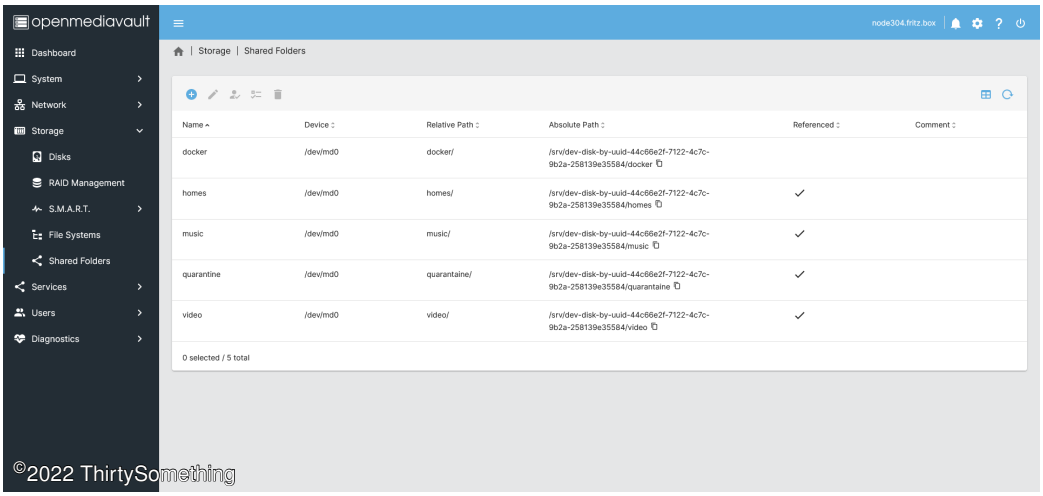

Figure 6: The OMV shared folders

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.

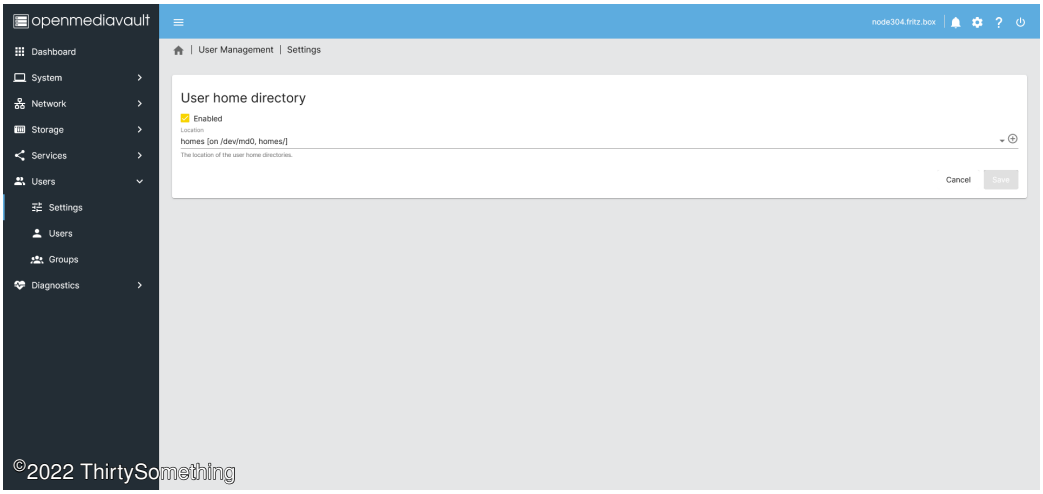

Figure 7: The OMV users home directory

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.

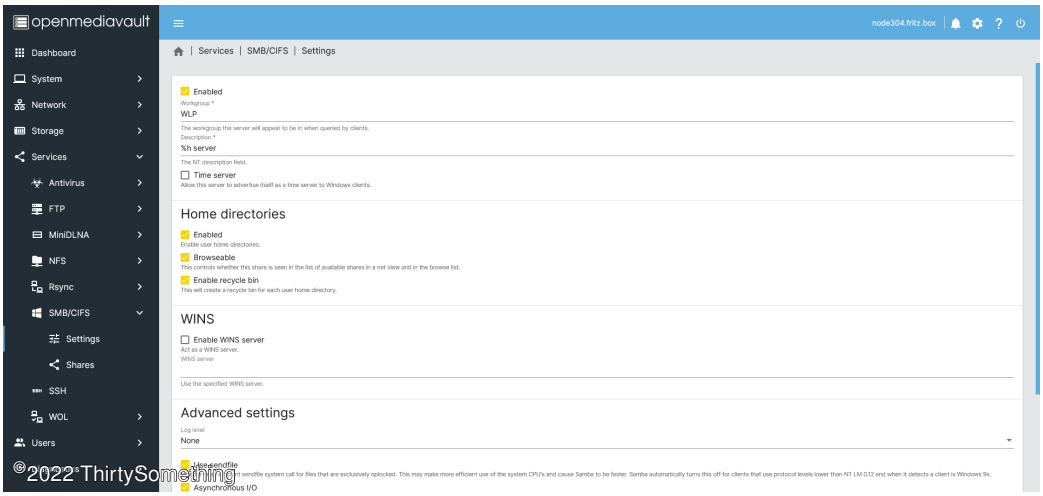

Figure 8: The OMV CIFS settings

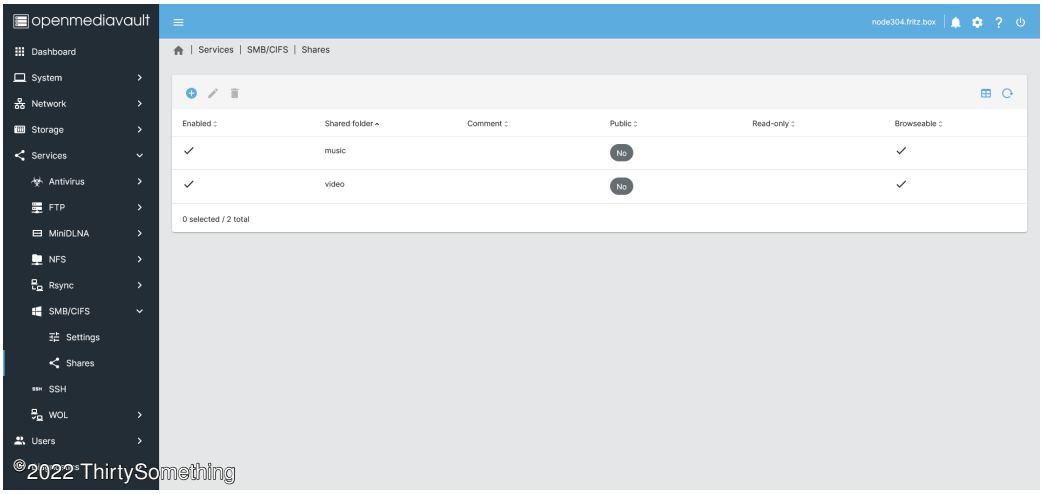

Figure 9: The OMV CIFS shares

As you can see there is actualy no public access to this shares. This means that only priviledged users can access them.

4.8 Tuning

4.8.1 SMB/CIFS

Transfering 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:

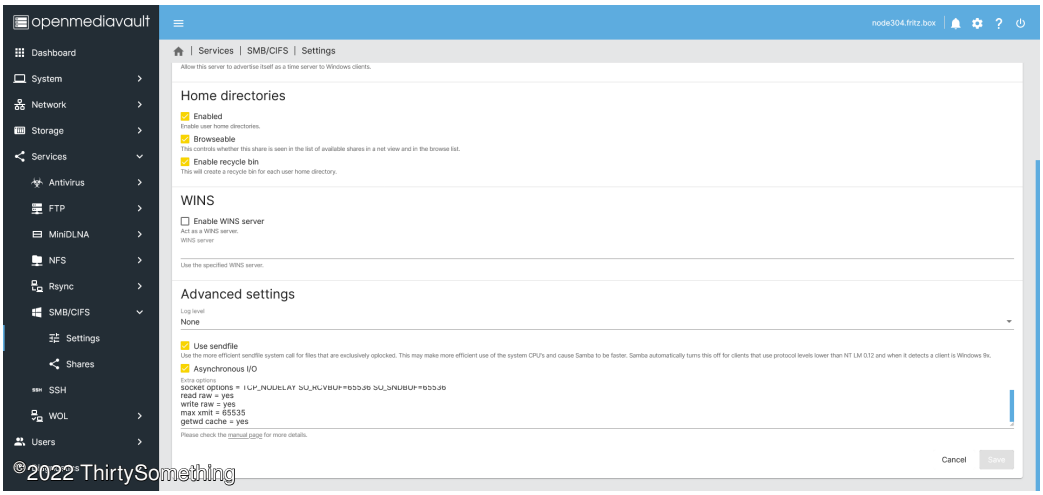

Figure 10: The OMV advanced CIFS settings

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.

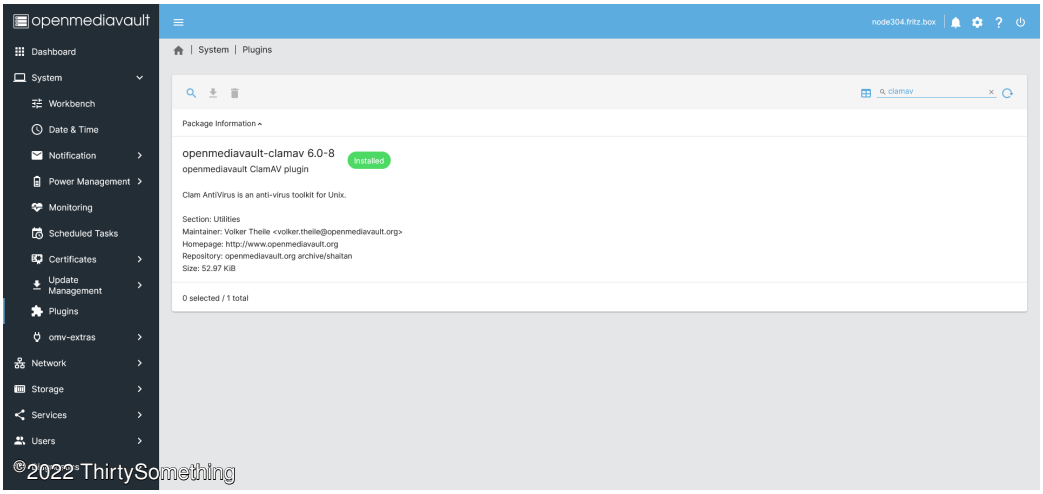

Figure 11: The OMV ClamAV plugin

In the setup we use the previously defined quarantine folder.

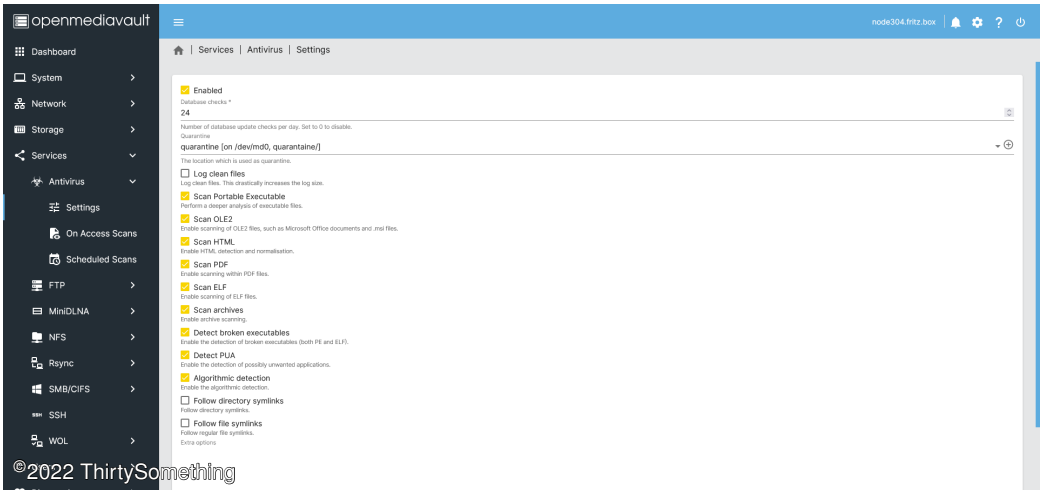

Figure 12: The antivirus settings

I enabled a scan on access for specific folders.

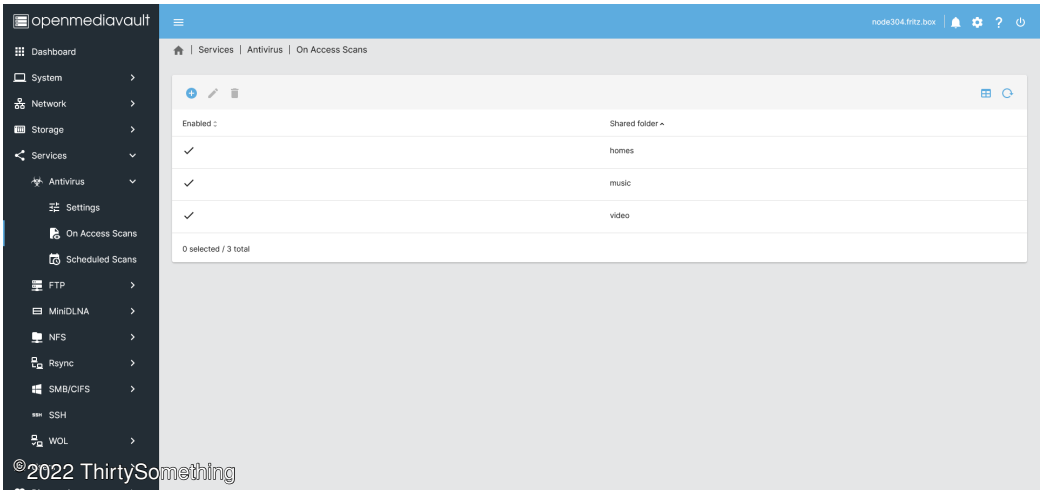

Figure 13: The antivirus on access scan

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

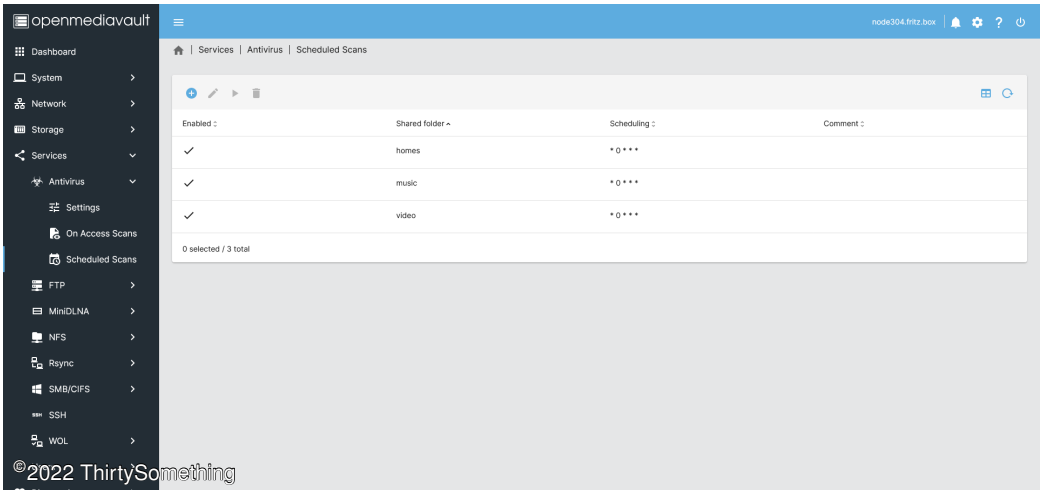

Figure 14: The antivirus scheduled scan

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`.

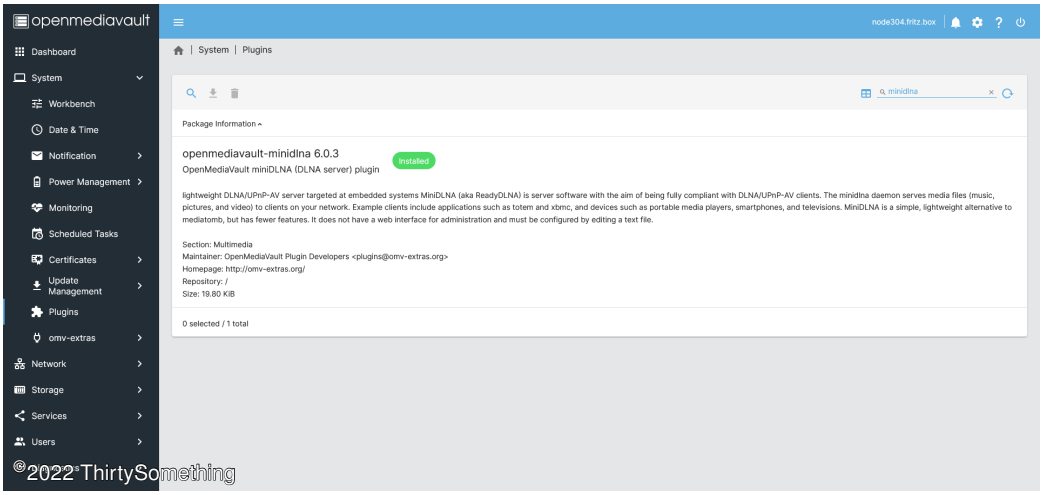

Figure 16: The MiniDLNA plugin

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

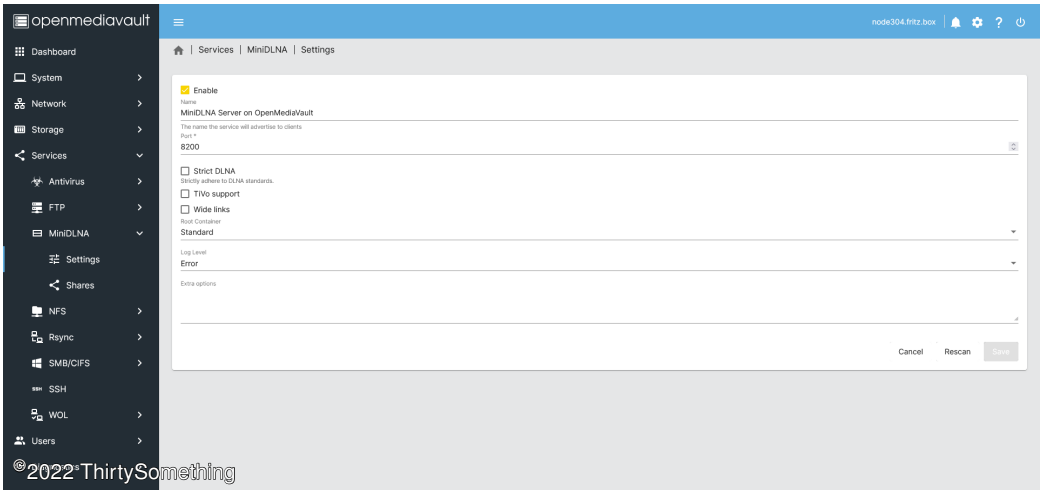

Figure 17: The MiniDLNA settings

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

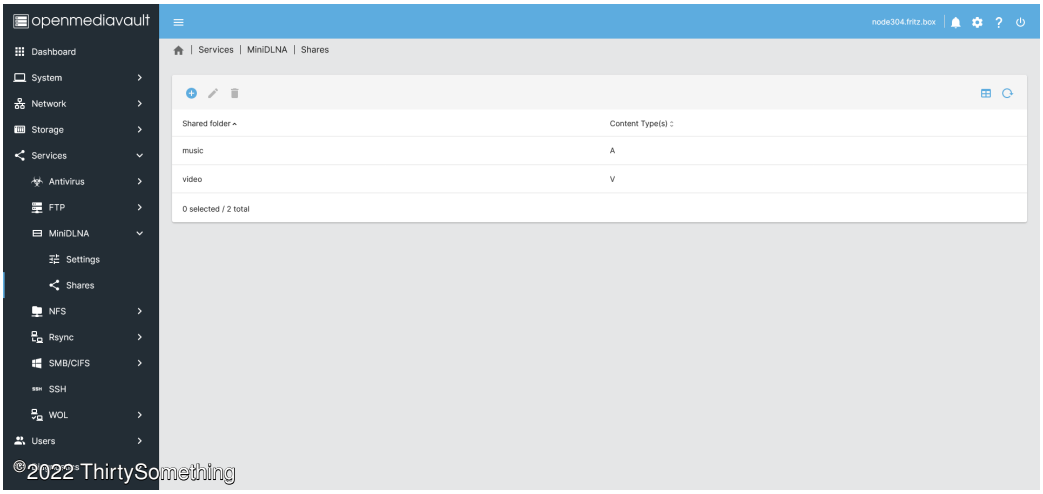

Figure 18: The MiniDLNA shares

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.

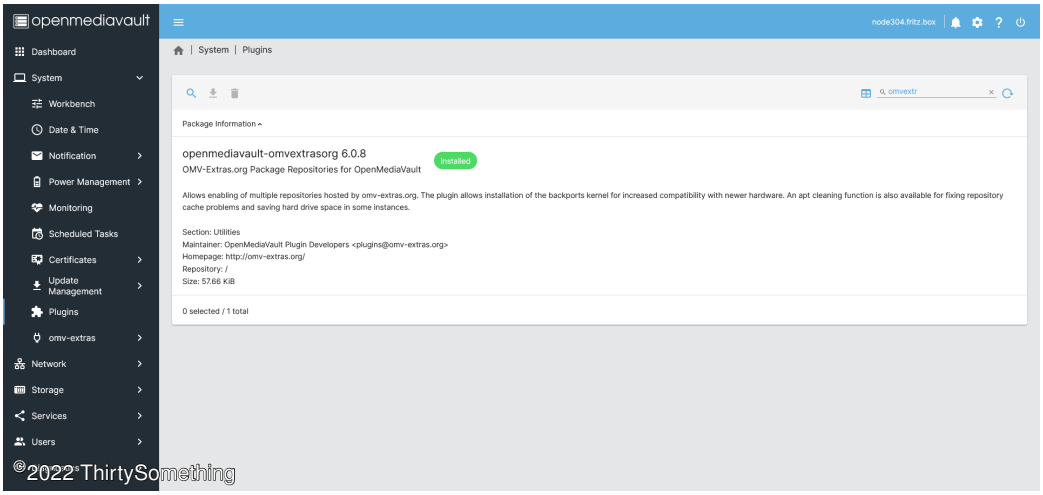

Figure 20: The OMV extras plugin

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.

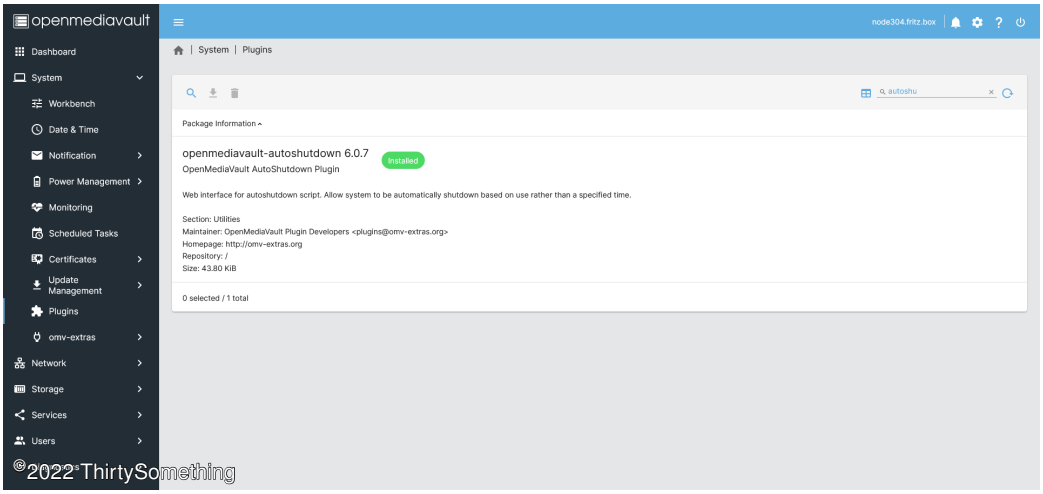

Figure 21: The OMV autoshutdown plugin

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

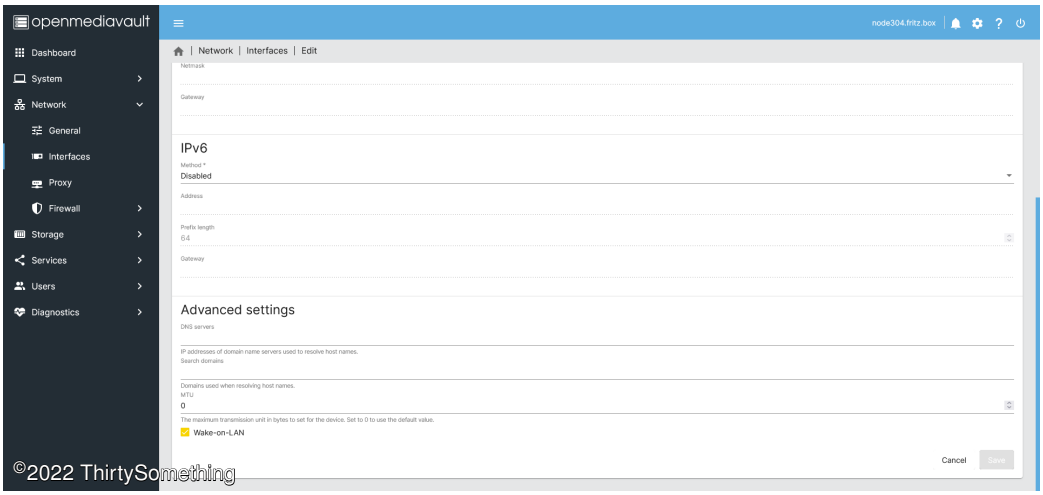

Figure 22: The OMV network card setup

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.

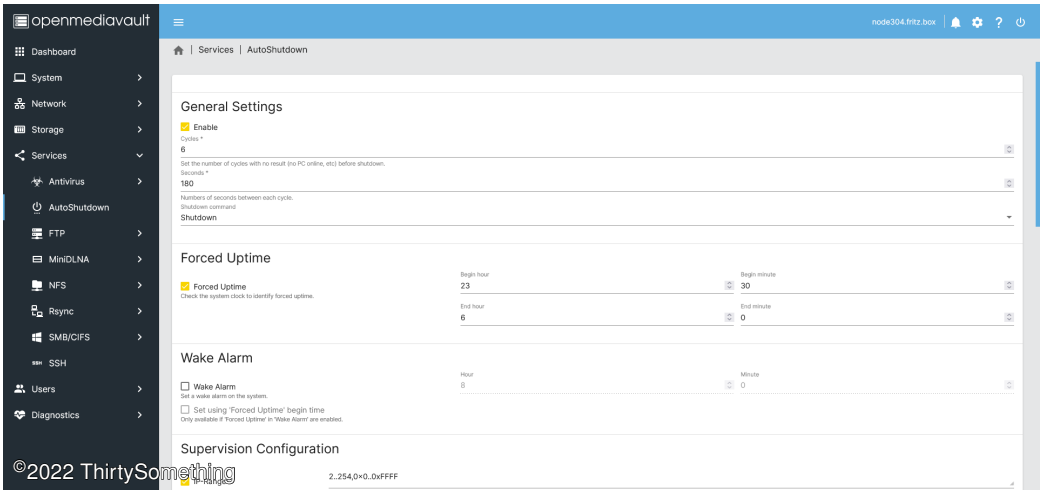

Figure 23: The autoshutdown setup 1

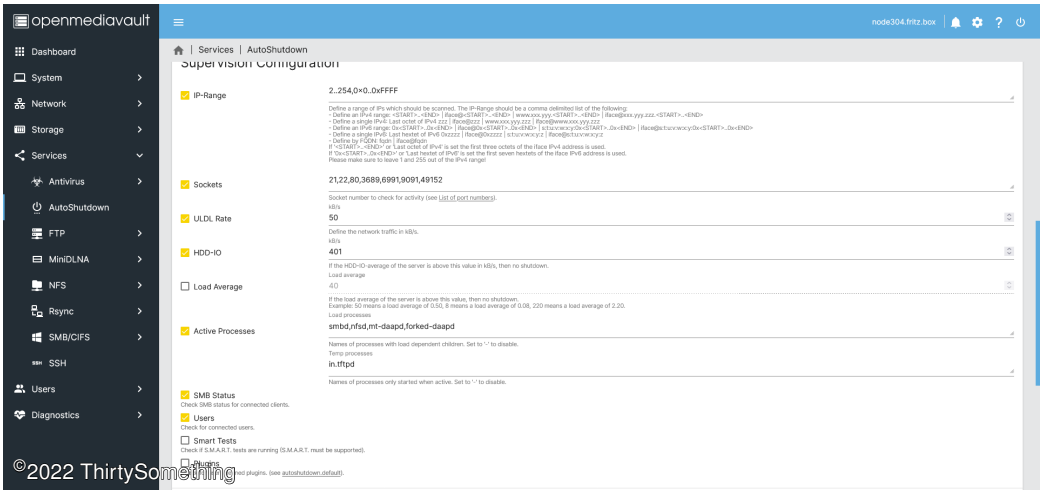

Figure 24: The autoshutdown setup 2

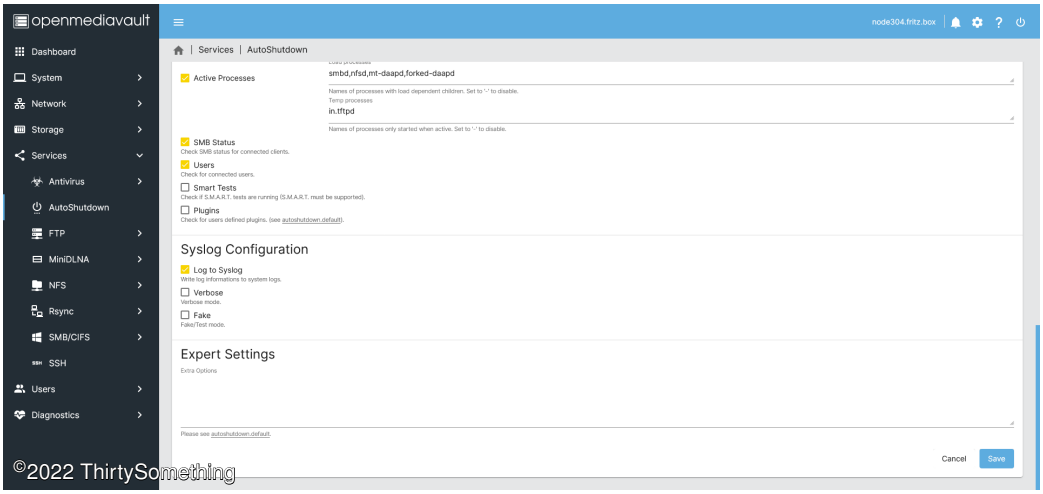

Figure 25: The autoshutdown setup 3

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.

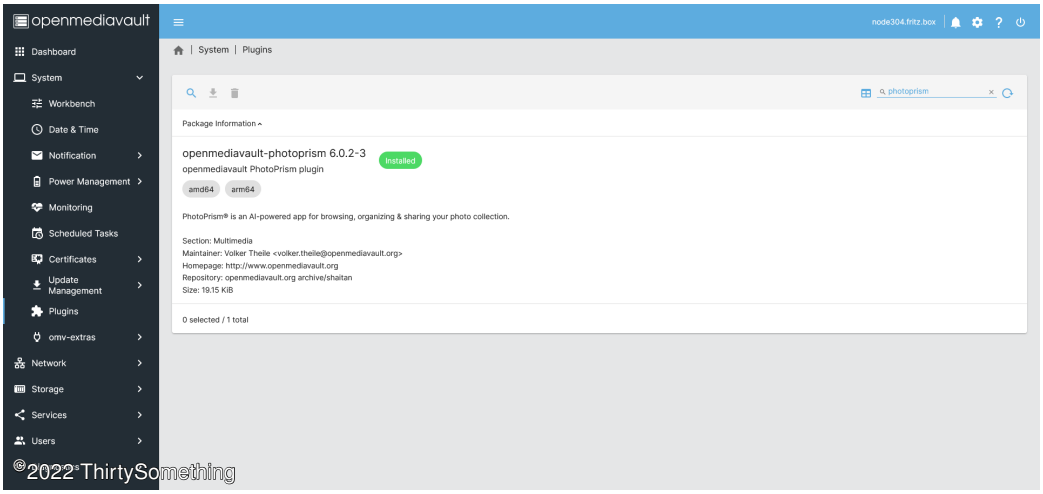

Figure 26: The OMV PhotoPrism plugin

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.

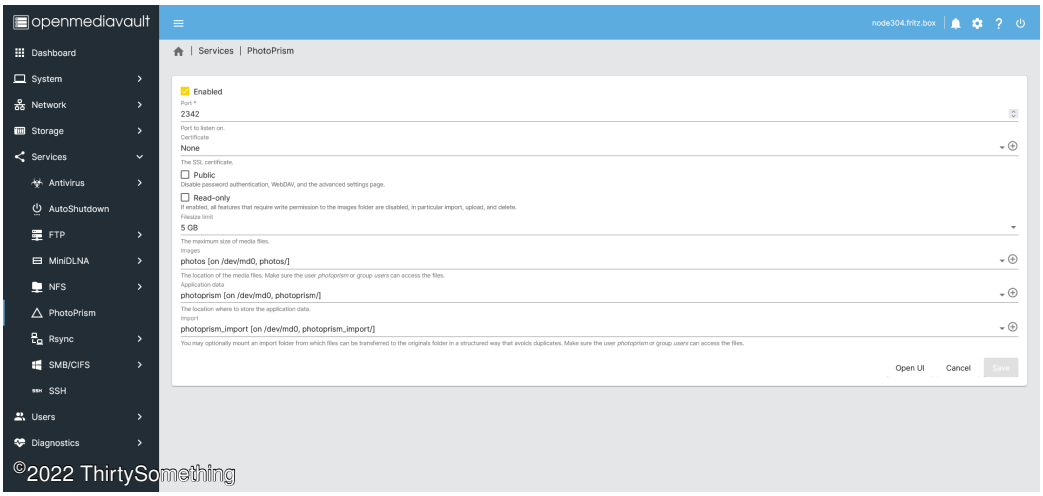

Figure 27: The OMV PhotoPrism setup

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. Allthough 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.

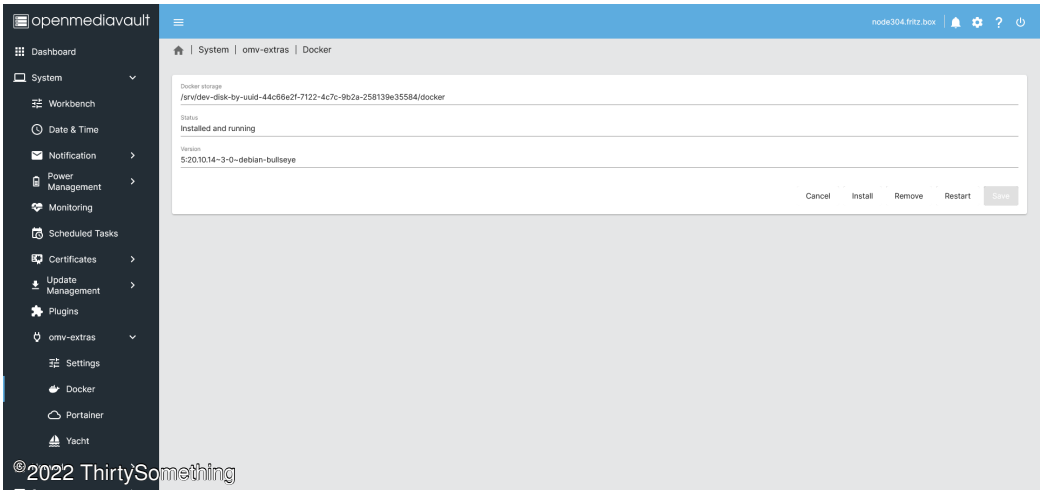

Figure 29: The Docker setup

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.

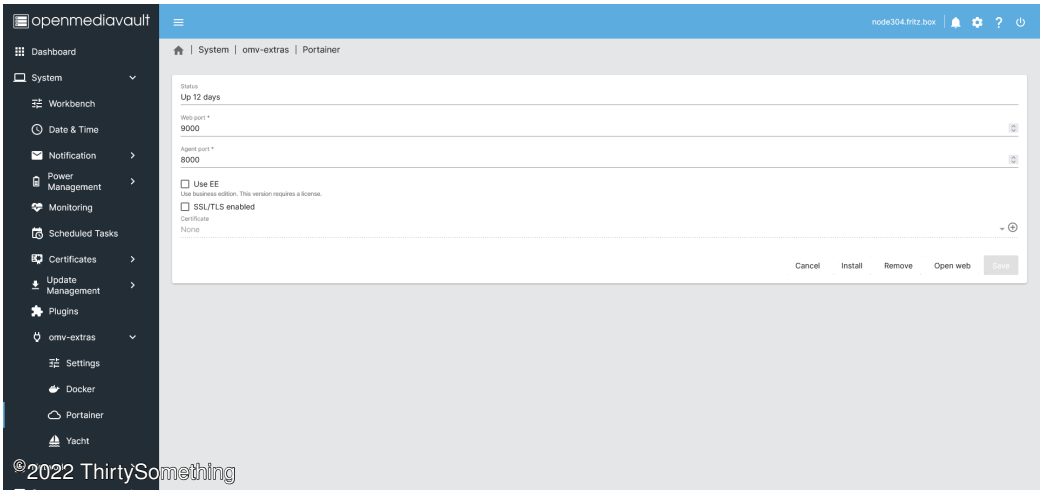

Figure 30: The Portainer setup

After installation Portainer is up and running.

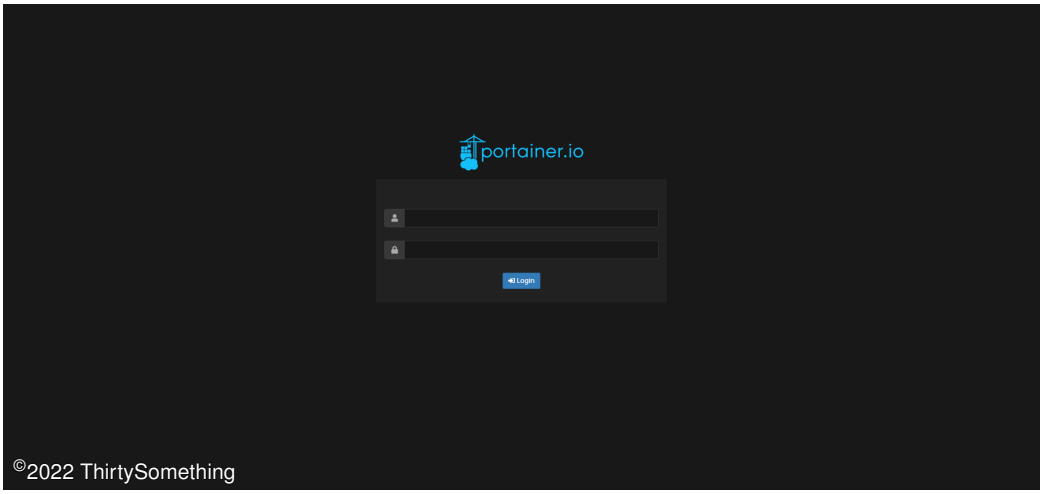

Figure 31: The Portainer login

6.3 Container list

- ~~Portainer – UI for docker~~ – done with docker installation.
- ~~SCM-Manager – SCM with git, svn and hg~~ – done.
- ~~MariaDB – Database for various projects~~ – done.
- ~~phpMyAdmin – UI for the database~~ – done.
- ~~Mosquitto – A MQTT broker~~ – done.
- ~~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 allthough 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 `docker-compose.yaml` for SCM-Manager looks like:

```
version: "3.8"
services:
  scmmanager:
    image: cloudogu/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:
```

The startup of the container should work without any problems.

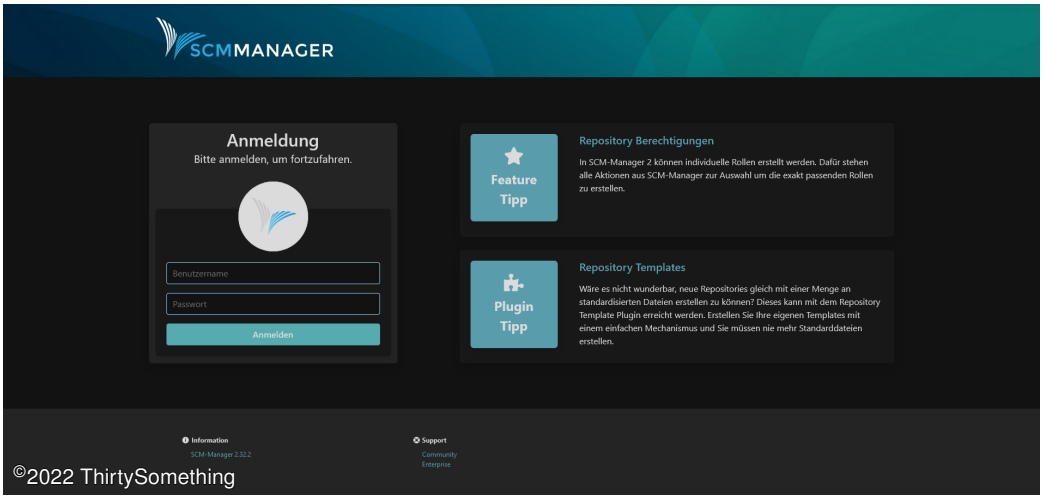

Figure 33: The SCM-Manager login

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 34: 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.

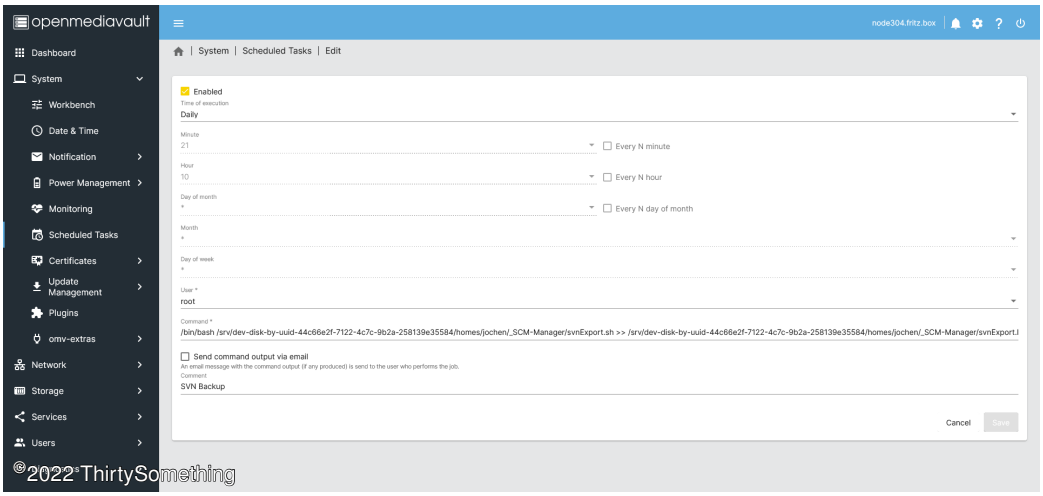

Figure 35: The SCM-Manager backup task

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. The `docker-compose.yaml` for Syncthing looks like:

```
version: "3.8"
services:
  syncthing:
    image: syncthing/syncthing:latest
    restart: unless-stopped
    environment:
      - PUID=1000
      - GUID=1000
    ports:
      - 8384:8384
      - 21027:21027/udp
      - 22000:22000/tcp
      - 22000:22000/udp
```

```
    volumes:
      - syncthing-data:/var/syncthing
volumes:
  syncthing-data:
```

Setting up the synchronized folders is as usual with Syncthing. The result may look like this one.

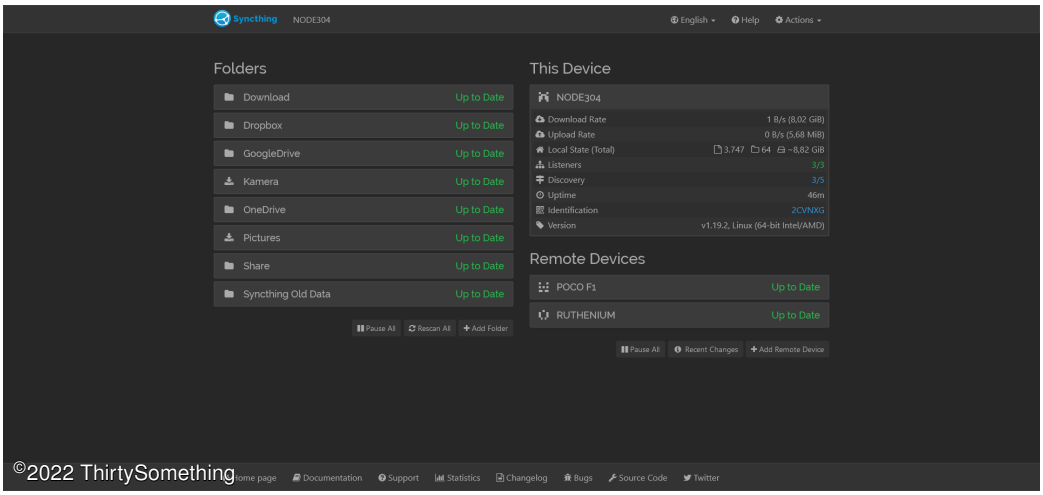

Figure 36: Syncthing

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. The `docker-compose.yaml` for Watchtower looks like:

```
version: "3.8"
services:
  watchtower:
    image: containrrr/watchtower
    restart: unless-stopped
    environment:
      # WATCHTOWER_MONITOR_ONLY: 'true'
      WATCHTOWER_NOTIFICATIONS: email
      WATCHTOWER_NOTIFICATION_EMAIL_FROM: <email>
      WATCHTOWER_NOTIFICATION_EMAIL_TO: <email>
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER: <smtp-server>
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PORT: <smtp-server-port>
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER_USER: <email>
      WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PASSWORD: <password>
      WATCHTOWER_NOTIFICATION_EMAIL_DELAY: 2
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

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

As developer there is a need for a database. I decided to use MariaDB. The `docker-compose.yaml` for MariaDB looks like:

```
version: "3.8"
services:
  mariadb:
    image: mariadb:latest
    restart: unless-stopped
    environment:
      MARIADB_ROOT_PASSWORD: <password>
    ports:
      - 3306:3306
    volumes:
      - mariadb-data:/var/lib/mysql
volumes:
  mariadb-data:
```

6.9 phpMyAdmin

To administrate the database I'm using this UI. The `docker-compose.yaml` for phpMyAdmin looks like:

```
version: "3.8"
services:
  phpmyadmin:
    image: phpmyadmin:latest
    restart: unless-stopped
    environment:
      PMA_HOST: <IP of NAS>
    ports:
      - 8088:80
```

And here we go:

Figure 37: phpMyAdmin

6.10 Mosquitto

I used Mosquitto in some projects. For testing purposes I've used the public version of Mosquitto or installed locally one on my developer machine. Now it's time to run it on my NAS. The `docker-compose.yaml` for Mosquitto looks like:

```
version: "3.8"
services:
  mosquitto:
    image: eclipse-mosquitto:latest
    restart: unless-stopped
    environment:
      - PUID=1000
      - GUID=1000
    ports:
      - 1883:1883
      - 9001:9001
    volumes:
      - mosquitto-config:/mosquitto/config
      - mosquitto-data:/mosquitto/data
      - mosquitto-log:/mosquitto/log
volumes:
  mosquitto-config:
  mosquitto-data:
  mosquitto-log:
```

6.11 Gerbera

Gerbera is a media server similar to ReadyMedia aka MiniDLNA. MiniDLNA is already available as a native plugin of OMV. Unfortunately it

happens twice that the logfile of this media server bloated up and occupied all free space on the OS drive. So I decided to check out an alternative. The `docker-compose.yaml` for Gerbera looks like:

```
version: "3.8"
services:
  gerbera:
    image: gerbera/gerbera:latest
    restart: unless-stopped
    environment:
      - UID=1000
      - GID=1000
    ports:
      - 49494:49494
    volumes:
      - gerbera-audio:/media/audio
      - gerbera-photo:/media/photo
      - gerbera-video:/media/video
volumes:
  gerbera-audio:
    driver: local
    driver_opts:
      type: "cifs"
      device: "//<IP of NAS>/music"
      o: "username=<username>,password=<password>,vers=3.0,uid=1000,gid=1000"
  gerbera-photo:
    driver: local
    driver_opts:
      type: "cifs"
      device: "//<IP of NAS>/photos"
      o: "username=<username>,password=<password>,vers=3.0,uid=1000,gid=1000"
  gerbera-video:
    driver: local
    driver_opts:
      type: "cifs"
      device: "//<IP of NAS>/video"
      o: "username=<username>,password=<password>,vers=3.0,uid=1000,gid=1000"
```

6.12 All-In-One

It is also possible to have an *All-In-One* file with all containers used as one `docker-compose.yaml` file. The `docker-compose.yaml` for All-In-One looks like:

```
version: "3.8"
services:
  # Media server
  gerbera:
    image: gerbera/gerbera:latest
    restart: unless-stopped
    environment:
      - UID=1000
      - GID=1000
    ports:
```

```
    - 49494:49494
  volumes:
    - gerbera-audio:/media/audio
    - gerbera-photo:/media/photo
    - gerbera-video:/media/video
# RDBMS
mariadb:
  image: mariadb:latest
  restart: unless-stopped
  environment:
    MARIADB_ROOT_PASSWORD: <password>
  ports:
    - 3306:3306
  volumes:
    - mariadb-data:/var/lib/mysql
# MQTT broker
mosquitto:
  image: eclipse-mosquitto:latest
  restart: unless-stopped
  environment:
    - PUID=1000
    - GUID=1000
  ports:
    - 1883:1883
    - 9001:9001
  volumes:
    - mosquitto-config:/mosquitto/config
    - mosquitto-data:/mosquitto/data
    - mosquitto-log:/mosquitto/log
# RDBMS admin tool
phpmyadmin:
  image: phpmyadmin:latest
  restart: unless-stopped
  environment:
    PMA_HOST: <IP of NAS>
  ports:
    - 8088:80
# Version control system
scmmanager:
  image: cloudogu/scm-manager:latest
  restart: unless-stopped
  ports:
    - 2222:2222
    - 8080:8080
  volumes:
    - scm-config:/var/cache/scm/work
    - scm-data:/var/lib/scm
# Container update utility
watchtower:
  image: containrrr/watchtower
  restart: unless-stopped
  environment:
    # WATCHTOWER_MONITOR_ONLY: 'true'
    WATCHTOWER_NOTIFICATIONS: email
    WATCHTOWER_NOTIFICATION_EMAIL_FROM: <email>
    WATCHTOWER_NOTIFICATION_EMAIL_TO: <email>
    WATCHTOWER_NOTIFICATION_EMAIL_SERVER: <smtp-server>
    WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PORT: <smtp-server-port>
    WATCHTOWER_NOTIFICATION_EMAIL_SERVER_USER: <email>
    WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PASSWORD: <password>
    WATCHTOWER_NOTIFICATION_EMAIL_DELAY: 2
  volumes:
```

```
      - /var/run/docker.sock:/var/run/docker.sock

volumes:
  # Media server
  gerbera-audio:
    driver: local
    driver_opts:
      type: "cifs"
      device: "//<IP of NAS>/music"
      o: "username=<username>,password=<password>,vers=3.0,uid=1000,gid=1000"
  gerbera-photo:
    driver: local
    driver_opts:
      type: "cifs"
      device: "//<IP of NAS>/photos"
      o: "username=<username>,password=<password>,vers=3.0,uid=1000,gid=1000"
  gerbera-video:
    driver: local
    driver_opts:
      type: "cifs"
      device: "//<IP of NAS>/video"
      o: "username=<username>,password=<password>,vers=3.0,uid=1000,gid=1000"
  # RDBMS
  mariadb-data:
  # MQTT broker
  mosquitto-config:
  mosquitto-data:
  mosquitto-log:
  # Version control system
  scm-config:
  scm-data:
```

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 this port.

Program	Port	Comment
Gerbera	49494	UI
MariaDB	3306	Database conneciton port
Mosquitto	1883	MQTT
Mosquitto	9001	MQTT
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
Mosquitto	1883	MQTT
SCM-Manager	2222	ssh
PhotoPrism	2342	UI
MariaDB	3306	Database connection port

SCM-Manager	8080	UI
phpMyAdmin	8088	UI
Portainer	9000	UI
Mosquitto	9001	MQTT
Syncthing	8384	UI
Syncthing	UPD:21027	Receive local discovery broadcasts
Syncthing	TCP:22000	Filetransfer
Syncthing	UDP:22000	Filetransfer
Gerbera	49494	UI

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_scm-data/_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 {
    svnlook 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 respoistory 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=$((0 + "${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 [[ "${COUNTER}" -lt "${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 "================================================================================"
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
    echo "--------------------------------------------------------------------------------"
done
echo ""
```

List of Figures

List of Tables

Glossary

All-In-One AIO is shorthand for All-In-One 30

ClamAV ClamAV, an open source antivirus software 13

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

Gerbera Gerbera, a free media server 29, 30, 33, 34

git git, a version control system 24, 25

MariaDB MariaDB, a RDBMS 28, 33

Mosquitto Mosquitto, an open source MQTT broker 29, 33, 34

NAS Network Attached Storage 3–5, 12, 15, 17, 22, 25, 29

OMV Open Media Vault, a NAS operating system 5, 7, 10, 13, 15, 22, 26, 29, 33

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

PhotoPrism PhotoPrism, AI powered photo manager 19, 33

phpMyAdmin phpMyAdmin, an UI for MariaDB 28, 33, 34

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

Portainer Portainer, a container management tool 17, 22, 23, 33, 34

SCM-Manager SCM-Manager, a version control system server 24–26, 33, 34

Subversion Subversion, a version control system 24, 25

Syncthing Syncthing, file synchronization 26, 27, 33, 34

Synology Manufacturer of NAS systems 3

Watchtower Watchtower, container management utility 27

WOL Wake On LAN, starts computer by specific network signal 17, 18