Bucket-NAS

Set up an encrypted NAS on Odroid-C2/C4 or Rasbperry 3/4 with OpenMediaVault

Jens Getreu

Revision History

Revision 2.6	16/03/23	JG

Table of Contents

This note explains how to set up a cheap Raid 1 NAS with an Odroid or Rasbperry board and two USB-harddisks using **OpenMediaVault**. Only very cheap and largely available hardware components are used. When very high availability is required it is recommended to hold available a second Oroid/Rasbperry board with a mirrored SD-card containing the NAS operating system. In this way the system has no single point of failure.

1. Hardware

Any Rasbperry like development board will do as long as Armbian[1] supports it. I prefer *Odroid-C2* over *Rasbperry 3* because of the better performance, larger memory (2GB) and Gigabit-Ethernet. As neither Odroid-C2 nor Rasbperry-3 have USB3 ports cheap USB2 hard-disks will do perfectly.

Prerequisites for Odroid:

- 1 *Odroid-C2*
- 1 SD-card UHS Speed Class 1 (U1), 4GB
- 1 USB 2 or USB 3 hard-disk containing your data, e.g. films, photos …
- 1 empty USB 2 or USB 3 hard-disk with the same size or bigger.

For optimal speed it is recommended[2] to use the separated USB host port for the second HDD via USB OTG host port on C2.

Prerequisites for Rasbperry:

- 1 *Rasbperry 3*
- 1 SD-card Class 10, 4GB
- 1 USB 2 or USB 3 hard-disk containing your data, e.g. films, photos …
- 1 empty USB 2 or USB 3 hard-disk with the same size or bigger.

[1] http://www.armbian.com/download/
[2] http://forum.odroid.com/viewtopic.php?f=140&t=22212

> You can connect up to 5 USB2 disks as the *Odroid-C2* has 5 USB2 ports. For very high availability you might also want to buy a second spare *Odroid/Rasbperry* with a second mirrored SD-card as instant replacement.

Designed as a project for my students I installed all devices in a 3€ paint-bucket. This is how it looks like without cover.

2. Copy Debian 9 on a micro SD card and initial setup

I recommend the Armbian distribution because it has good hardware support and is available for wide range of Rasbperry-like boards.

See also the official documentation[3].

2.1. Download image

Download: Odroid C2 – Armbian[4] or Alternative version[5]

[3] https://docs.armbian.com/

[4] https://dl.armbian.com/odroidc2/Debian_stretch_default.7z

2.2. Copy image on SD-card

```
> apt-get install p7zip
> mkdir tmp; cd tmp
> 7z x Armbian_5.47_Odroidc2_Debian_stretch_default_3.16.57.7z
> dd if=Armbian_5.47_Odroidc2_Debian_stretch_default_3.16.57.img of=/dev/
mmcblk0 bs=8M
```

[6]

Insert the SD-card in your Odroid or Rasbperry and power-on.

2.3. Login

Login as root on console or via SSH and use password 1234. Change default password. I will call it "<naspassword>" later.

2.4. Console stuff

```
> apt-get update
> apt-get upgrade
> dpkg-reconfigure tzdata
> apt-get install mc aptitude cryptsetup keyboard-configuration
> dpkg-reconfigure keyboard-configuration
> service keyboard-setup restart
> update-initramfs -u
```

3. Install OpenMediaVault

3.1. Armbian’s configuration tool

Armbian comes with a configuration tool `armbian-config` automating the installation of **OpenMediaVault**.

```
> armbian-config
```

In the ncurses-application choose: `Software` → `Softy` → `OMV` for **OpenMediaVault**.

[5] https://www.armbian.com/odroid-c2/

[6] Convention: Lines starting with `>` should be executed as root in a bash shell.

An alternative setup guide can be found here[7].

3.2. Manual install

The following section shows how to install **OpenMediaVault** manually.

1. Set up link to OVM repositiory

   Create the file `/etc/apt/sources.list.d/openmediavault.list` with the following content:

```
deb http://packages.openmediavault.org/public arrakis main
# deb http://downloads.sourceforge.net/project/openmediavault/packages
 arrakis main
## Uncomment the following line to add software from the proposed
 repository.
# deb http://packages.openmediavault.org/public arrakis-proposed main
# deb http://downloads.sourceforge.net/project/openmediavault/packages
 arrakis-proposed main
## This software is not part of OpenMediaVault, but is offered by
 third-party
## developers as a service to OpenMediaVault users.
# deb http://packages.openmediavault.org/public arrakis partner
# deb http://downloads.sourceforge.net/project/openmediavault/packages
 arrakis partner
```

2. Set up your keys

```
> wget -O - http://packages.openmediavault.org/public/archive.key |
 apt-key add -
```

3. Install

```
> apt-get update
> apt-get install openmediavault openmediavault-keyring
> omv-initsystem
```

4. Install ovmextras (optional)

```
> wget http://omv-extras.org/openmediavault-
omvextrasorg_latest_all4.deb
```

[7] https://forum.openmediavault.org/index.php/Thread/21234-Install-OMV4-on-Debian-9-Stretch/

```
> dpkg -i openmediavault-omvextrasorg_latest_all4.deb
> apt-get update
```

5. Reboot

> In order to reconnect via `slogin` you need to enable SSH in the OpenmediaVault Web-GUI `Services # SSH # Enable`

```
> reboot
```

3.3. Login in

web interface

```
> firefox https://bucketnas.lan # <- replace with your NAS ip here
```

- username = admin
- password = openmediavault

console/ssh

```
> slogin -l root bucketnas.lan # <- replace with your NAS ip here
```

- username = root
- password = <naspassword>

3.4. Install the Luksencryption plugin (recommended)

The `openmediavault-luksencryption` plugin is very handy to lock or unlock the encryption layer.

Install in console:

```
> apt-get install openmediavault-luksencryption
```

or Web-GUI:

```
openmediavault web-gui -> System -> Plugins -> openmediavault-
luksencryption
```

3.5. Install plugins (optional)

Interesting modules:

```
> apt-get install openmediavault-webdav openmediavault-minidlna
```

[8]

3.6. Enable ssl/tls connections (https:…)

Open Web-GUI and

```
openmediavault web-gui -> System -> Certificats -> ssl -> add -> create
```

```
openmediavault web-gui -> General Settings -> Web administration -> secure
 connection
```

then enable SSL/TLS and choose the above created certificate.

3.7. Change passwords

1. Web-GUI-adminstrator password

   ```
   firefox https://bucketnas.lan # <- replace with your NAS ip here
   ```

   Username: *admin*, initial password: *openmediavault*

   ```
   OpenMediaVault -> General Settings -> Web Administrator Password
   ```

2. Root password

   ```
   > slogin -l root bucketnas.lan # <- replace with your NAS ip here
   ```

   Initial password is: *odroid*

   ```
   > passwd
   ```

[8] NOTE: I could not get `openmediavault-webdav` working with *OpenMediaVault 3.0.13*.

4. Create an empty Raid 1 and copy your data

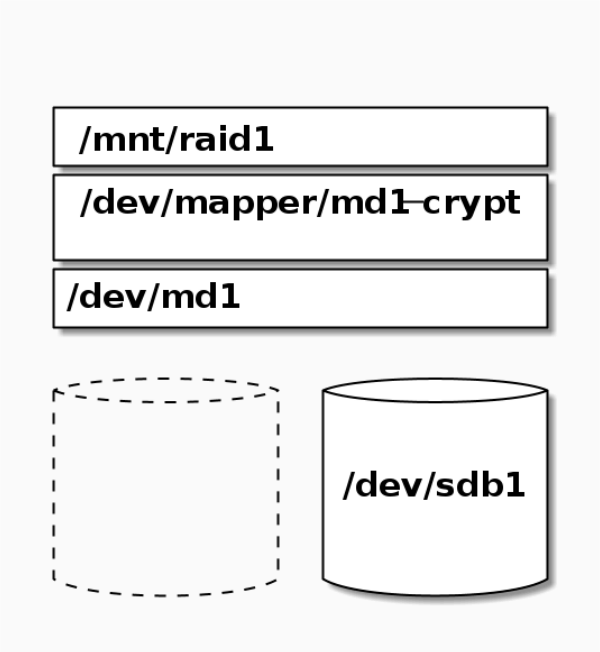


Figure 1. Empty degraded Raid 1

4.1. Prepare an empty degraded Raid 1 with one disk

1. Create partition on empty disk

   It is highly recommended to pre-partition the disks to be used in the array. Since most RAID users are selecting HDDs >2 TB, GPT partition tables are required and recommended. Disks are easily partitioned using gptfdisk.

   - After created, the partition type should be assigned hex code `FD00`.
   - If a larger disk array is employed, consider assigning disk labels or partition labels to make it easier to identify an individual disk later.
   - Creating partitions that are of the same size on each of the devices is preferred.
   - A good tip is to leave approx 100 MB at the end of the device when partitioning. See below for rationale.

     > It is also possible to create a RAID directly on the raw disks (without partitions), but not recommended because it can cause problems when swapping a failed disk. When replacing a failed

> disk of a RAID, the new disk has to be exactly the same size as the failed disk or bigger — otherwise the array recreation process will not work. Even hard drives of the same manufacturer and model can have small size differences. By leaving a little space at the end of the disk unallocated one can compensate for the size differences between drives, which makes choosing a replacement drive model easier. Therefore, it is good practice to **leave about 100 MB of unallocated space at the end of the disk.** [9]

```
> fdisk /dev/sdb
```

2. Create empty degraded Raid 1

```
> mdadm --create /dev/md1 --level=1 --raid-devices=2 missing /dev/sdb1
```

[10]

3. Create encryption layer on top of our Raid

```
> cryptsetup -v luksFormat /dev/md1
```

4. Open encryption layer

```
cryptsetup luksOpen /dev/md1 md1-crypt
Enter passphrase for /dev/md1
```

5. Create filesystem

```
> mkfs.ext4 /dev/mapper/md1-crypt
```

6. Mount this disk as destination disk

```
> mkdir /mnt/raid1
> mount -t ext4 /dev/mapper/md1-crypt /mnt/raid1
```

[9] https://wiki.archlinux.org/index.php/RAID#Create_the_Partition_Table_.28GPT.29

[10] In case you have to remount to degraded array later do: `mdadm -A --run /dev/md1 /dev/sdb1`

4.2. Copy your data onto the new disk

1. Mount disk with existing data

```
> mkdir /mnt/from
> mount -t ext4 /dev/sda1 /mnt/from
```

2. Copy

```
> cd /mnt/from
> cp -vur * /mnt/raid1
```

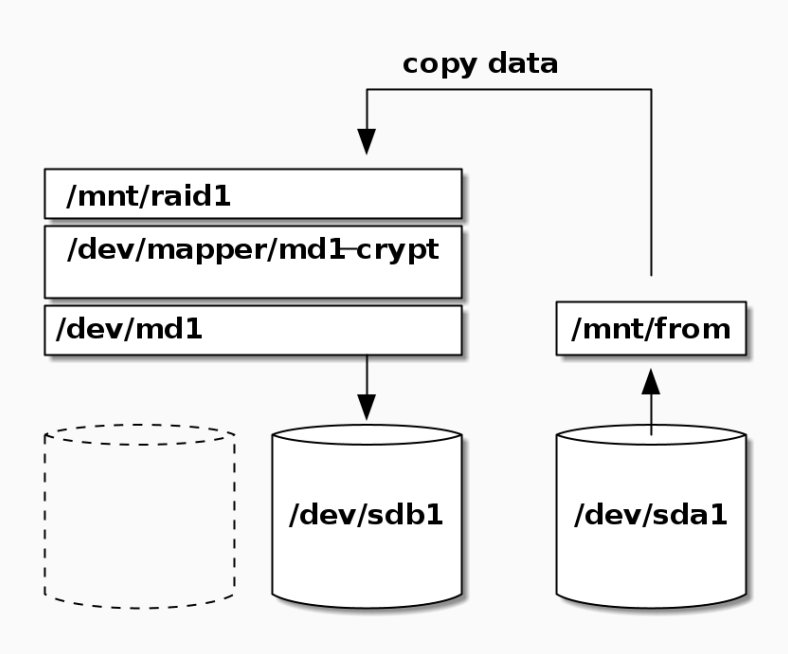


Figure 2. Copying

3. Unmount

```
> umount /mnt/from
> umount /mnt/raid1
```

4. Create /etc/fstab entry

   Mount the Raid via the Web-GUI. It will leave an entry in `/etc/fstab` [11]. From now on the filesystem should be mounted automatically as soon as you decrypt.

[11] The line looks like this: `/dev/mapper/openmediavault-crypt on /media/30abbf98-075e-420c-a8c4-dbccd3f7b60d type ext4`

4.3. Prepare a second empty disk

After having checked that all your data is well copied on your (still degraded) Raid 1, the original disk is now free and you can use it to complete your Raid 1.

Partition `/dev/sda1`:

> This deletes the content of `/dev/sda1`. Make sure that all your data was previously correctly copied on `/dev/raid1`!
>
> The new partition `/dev/sda1` must have exactly the same size than `/dev/sdb1` (or bigger).

```
> fdisk /dev/sda
```

[12]

4.4. Recover and sync

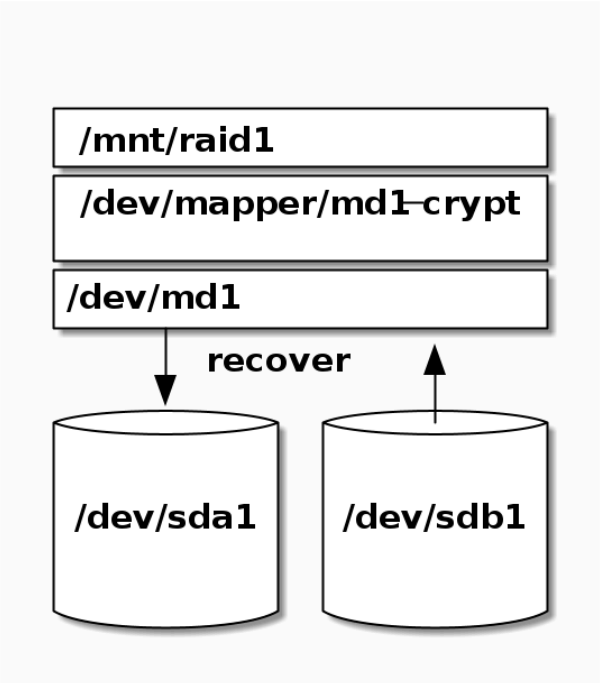


Figure 3. Complete Raid 1, recover and sync

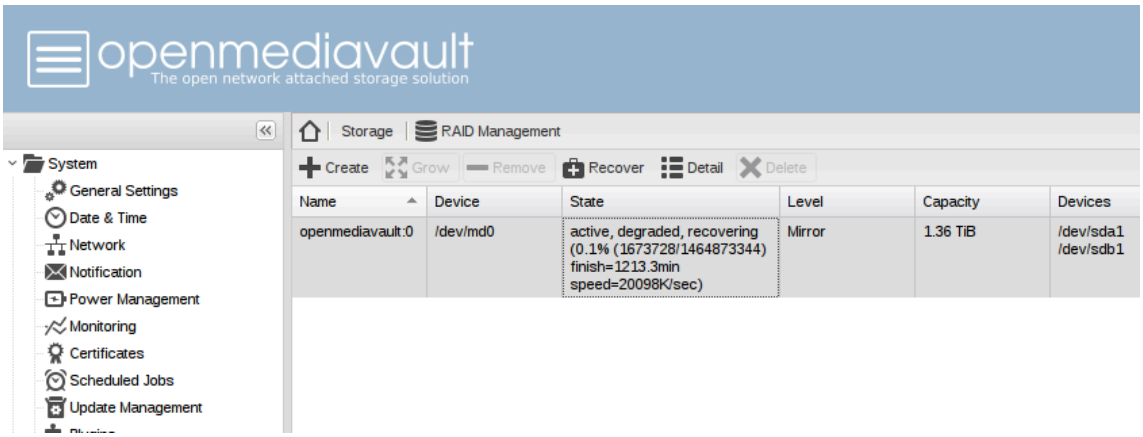
`(rw,noexec,relatime,data=ordered,jqfmt=vfsv0,usrjquota=aquota.user,grpjquota=aquota.group)` (your UUID is different).

[12] In the first version of this document I recommend to create a file system after partitioning with `mkfs.ext4 /dev/sda1`. Since our new partition will be entirely encrypted, we do not need an filesystem at this stage, so no filesystem creation is required.

1. Start sync

```
> mdadm --add-spare /dev/md1 /dev/sda1
> mdadm --readwrite /dev/md1
> cat /proc/mdstat
  Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5]
 [raid4] [raid10]
  md1 : active raid1 sdb1[2] sda1[0]
        2929928640 blocks super 1.2 [2/1] [U_]
        [=>...................]  recovery =  5.4%
 (160290304/2929928640) finish=2286.7min speed=20185K/sec
        bitmap: 0/22 pages [0KB], 65536KB chunk
```

As you can see on the following screenshot the synchronisation of 1.36 TB takes 19h and the speed is approx. 20MB/sec which is a very good value for writing USB 2 disks.



Encryption does not have any impact on the speed with our chosen hardware.

5. Start and stop the system

5.1. Stop

Login as root:

```
> shutdown
```

5.2. Start

In order to populate the internal OpenMediaVault database propertly, you must mount the filesystem once using the OVM's webgui in the following order:

1. Assemble Raid

```
OpenMediaVault -> Storage -> Software RAID
```

2. Decrypt

```
OpenMediaVault -> Storage -> Encryption
```

3. Mount file system

```
OpenMediaVault -> Storage -> File Systems
```

4. Select folders to share

```
OpenMediaVault -> Storage -> Shared Folders
```

From now on, you can use the shell to automate the above process when you restart the system. Make sure to always reference the same mount point that was generated automatically by OVM in step 3.

5.3. Sample shell start session

The following shell log shows how to start the system after reboot.

Assemble RAID1

```
root@bucketnas:~/bin# mdadm --detail --scan
root@bucketnas:~/bin#

root@bucketnas:~/bin# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda           8:0    0   2.7T  0 disk
##sda1        8:1    0   2.7T  0 part
sdb           8:16   0   2.7T  0 disk
##sdb1        8:17   0   2.7T  0 part

root@bucketnas:~/bin# mdadm -A /dev/md1 /dev/sda1 /dev/sdb1
mdadm: /dev/md1 has been started with 2 drives.

root@bucketnas:~/bin# mdadm --detail --scan
ARRAY /dev/md1 metadata=... name=bucketnas:1 UUID=0ed40ecb-4c61-41d2-b3a3-
f62892324db8/
```

```
root@bucketnas:~/bin# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda           8:0    0   2.7T  0 disk
##sda1        8:1    0   2.7T  0 part
  ##md1       9:1    0   2.7T  0 raid1
sdb           8:16   0   2.7T  0 disk
##sdb1        8:17   0   2.7T  0 part
  ##md1       9:1    0   2.7T  0 raid1
```

Decrypt

```
root@bucketnas:~/bin# cryptsetup open --type luks /dev/md1 md1-crypt
Enter passphrase for /dev/md1:

root@bucketnas:~/bin# ls /dev/mapper
control  md1-crypt
```

Mount decrypted filesystem

```
# Replace the mount point `/srv/dev-disk-by-uuid-0ed40e...` with the one
 that was generated by OMV during the first mount via GUI.
root@bucketnas:~/bin# mount -t ext4 -o rw,noexec,relatime /dev/mapper/md1-
crypt /srv/dev-disk-by-uuid-0ed40ecb-4c61-41d2-b3a3-f62892324db8/

root@bucketnas:~/bin# mount |grep mapper
/dev/mapper/md1-crypt on /srv/dev-disk-by-uuid-0ed40ecb-4c61-41d2-b3a3-
f62892324db8 type ext4 (rw,noexec,relatime)
```

Restart your sevices

```
root@bucketnas:~/bin# service minidlna restart
```

5.4. Sample session: close filesystem

Vain attempt

```
root@bucketnas:~/bin# cryptsetup close md1-crypt
Device md1-crypt is still in use.

root@bucketnas:~/bin# cryptsetup status md1-crypt
/dev/mapper/md1-crypt is active and is in use.
  type:    LUKS2
  cipher:  ...
  keysize: ...
  key location: ...
```

```
  device:  /dev/md1
  sector size:  ...
  offset:  ... sectors
  size:    ... sectors
  mode:    read/write
```

Unount filesystem

```
root@bucketnas:~/bin# mount | grep /dev/mapper
/dev/mapper/md1-crypt on /srv/dev-disk-by-
uuid-0ed40ecb-4c61-41d2-b3a3-f62892324db8 type ext4
 (rw,relatime,jqfmt=vfsv0,usrjquota=aquota.user,grpjquota=aquota.group)

root@bucketnas:~/bin# umount /dev/mapper/md1-crypt

root@bucketnas:~/bin# mount | grep /dev/mapper
root@bucketnas:~/bin#
```

Close slot

```
root@bucketnas:~/bin# cryptsetup close md1-crypt

root@bucketnas:~/bin# cryptsetup status md1-crypt
/dev/mapper/md1-crypt is inactive.
```

Raids are reboot resitant. There is usually no need to stop them. The following is documented for completeness. Omit!

Disassemble RAID1

```
root@bucketnas:~/bin# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda           8:0    0   2.7T  0 disk
##sda1        8:1    0   2.7T  0 part
  ##md1       9:1    0   2.7T  0 raid1
sdb           8:16   0   2.7T  0 disk
##sdb1        8:17   0   2.7T  0 part
  ##md1       9:1    0   2.7T  0 raid1

root@bucketnas:~/bin# mdadm --detail --scan
ARRAY /dev/md1 metadata=... name=bucketnas:1
 UUID=b1feed48:13bd99ea:dc0c4ffb:69680134

root@bucketnas:~/bin# mdadm -S /dev/md1
mdadm: stopped /dev/md1
```

```
root@bucketnas:~/bin# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda           8:0    0   2.7T  0 disk
##sda1        8:1    0   2.7T  0 part
sdb           8:16   0   2.7T  0 disk
##sdb1        8:17   0   2.7T  0 part

root@bucketnas:~/bin# mdadm --detail --scan
root@bucketnas:~/bin#
```

5.5. Sample startscript

This script resides on your NAS. If you want to call it from your client, adapt it using `ssh` and `sshpass`.

Sample startscript (please adjust to your needs)

```
#!/bin/sh
# setup decryption layer (will prompt for passwd)
cryptsetup open --type luks $(ls /dev/md*|head -n 1)  md1-crypt

# mount filesystem
mount -t ext4 -o
 rw,noexec,relatime,data=ordered,jqfmt=vfsv0,usrjquota=aquota.user,grpjquota=aquota.gro
dev/mapper/md1-crypt /srv/dev-disk-by-uuid-0ed40ecb-4c61-41d2-b3a3-
f62892324db8

# restart services
service minidlna restart

# create log directory, otherwise service will not start
mkdir -p /var/log/nginx
service nginx restart
```

5.6. Troubleshooting

Logical volume is not available

Problem

One Raid member device is a logical volume and it is not available.

```
> lvdisplay
--- Logical volume ---
```

```
  LV Path                /dev/vg-sdb1-sdc1/lv-combined
  LV Name                lv-combined
  VG Name                vg-sdb1-sdc1
  LV Write Access        read/write
  LV Status              NOT available
  LV Size                2.73 TiB
  Current LE             715347
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
```

```
> lvchange -aay
```

or if link to device is available

```
> lvchange -ay /dev/vg-sdb1-sdc1/lv-combined
```

```
  --- Logical volume ---
  LV Path                /dev/vg-sdb1-sdc1/lv-combined
  LV Name                lv-combined
  VG Name                vg-sdb1-sdc1
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                2.73 TiB
  Current LE             715347
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

Only one Raid member is started

Problem

Raid starts degraded and you see and error message like `/dev/md1 has been started with 1 drive (out of 2)`.

1. Check the member devices.

```
> mdadm -E /dev/sdb1
```

```
sdb1:
          Magic : a92b4efc
        Version : 1.2
    Feature Map : 0x1
   Super Offset : 8 sectors
   Unused Space : before=262056 sectors, after=1152 sectors
          State : clean
Internal Bitmap : 8 sectors from superblock
    Update Time : Tue Aug 16 20:52:39 2016
  Bad Block Log : 512 entries available at offset 72 sectors
       Checksum : 7c33262b - correct
         Events : 64894
```

```
> mdadm -E /dev/sda1

sda1:
          Magic : a92b4efc
        Version : 1.2
    Feature Map : 0x1
   Super Offset : 8 sectors
   Unused Space : before=262056 sectors, after=8 sectors
          State : clean
Internal Bitmap : 8 sectors from superblock
    Update Time : Tue Aug 16 23:00:46 2016
  Bad Block Log : 512 entries available at offset 72 sectors
       Checksum : 8c9eaaac - correct
         Events : 64898
```

2. Stop Raid.

```
> umount /dev/mapper/md1-crypt
> cryptsetup close md1-crypt
> mdadm -S /dev/md1
mdadm: stopped /dev/md1
```

3. Check Raid status.

```
> mdadm -D /dev/md1

/dev/md1:
        Version :
     Raid Level : raid0
  Total Devices : 0
```

```
          State : inactive
```

4. Assemble the members.

```
> mdadm -A /dev/md1 /dev/sda1 /dev/sdb1
mdadm: /dev/md1 has been started with 1 drive (out of 2).
```

5. Add the missing device.

```
> mdadm /dev/md1 --add /dev/sdb1
mdadm: re-added /dev/sdb1
```

6. Check Raid status.

```
> mdadm -D /dev/md1

/dev/md1:
        Version : 1.2
     Raid Level : raid1
          State : active
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0
         Events : 163202

 Number   Major   Minor   RaidDevice State
    2       8       17        0      spare rebuilding /dev/sdb1
    1       8        1        1      active sync /dev/sda1
```

If the above does not help try

```
> mdadm --stop /dev/md1
> mdadm --assemble --run --force --update=resync /dev/md1 /dev/sda1 /dev/
sdb1
> mdadm --readwrite /dev/md1
```

A detailed explanation can be found in this article[13].

[13] https://web.archive.org/web/20150116070025/http://sysadmin.blog.de:80/2011/11/08/resync-software-raid-erzwingen-12137452/

6. Secure your NAS with a firewall

To protect your **BucketNas** a firewall is recommended. My preferred configuration method is using the *Uncomplicated firewall (UFW)* configuration tool, because it configures IPv4 and IPv6 in one go.

6.1. Firewall configuration with UFW

The *Uncomplicated firewall (UFW)* is a configuration tool that abstracts a big deal of firewall complexity. Learn more here: An Introduction to Uncomplicated Firewall (UFW)[14]

Install UFW:

```
> apt-get install ufw
```

Most services bring their own ufw configuration, *Minidlna* doesn't. We have to add one. Create a file `/etc/ufw/applications.d/minidlna` with the following content:

```
[Minidlna]
title=Media server
description=MiniDLNA is server software with the aim of being fully
 compliant with DLNA/UPnP clients. The MiniDNLA daemon serves media files
 (music, pictures, and video) to clients on a network.
ports=1900/udp|8200/tcp
```

Activate ufw-profiles:

```
> ufw allow Minidlna
> ufw allow OpenSSH
> ufw allow Samba
> ufw allow "Nginx Full"
> ufw enable
```

Check activation. Below you see the expected results:

```
> ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
```

[14] https://www.linux.com/learn/introduction-uncomplicated-firewall-ufw

```
New profiles: skip

To                         Action      From
--                         ------      ----
1900/udp (Minidlna)        ALLOW IN    Anywhere
8200/tcp (Minidlna)        ALLOW IN    Anywhere
22/tcp (OpenSSH)           ALLOW IN    Anywhere
137,138/udp (Samba)        ALLOW IN    Anywhere
139,445/tcp (Samba)        ALLOW IN    Anywhere
80,443/tcp (Nginx Full)    ALLOW IN    Anywhere
1900/udp (Minidlna (v6))   ALLOW IN    Anywhere (v6)
8200/tcp (Minidlna (v6))   ALLOW IN    Anywhere (v6)
22/tcp (OpenSSH (v6))      ALLOW IN    Anywhere (v6)
137,138/udp (Samba (v6))   ALLOW IN    Anywhere (v6)
139,445/tcp (Samba (v6))   ALLOW IN    Anywhere (v6)
80,443/tcp (Nginx Full (v6)) ALLOW IN    Anywhere (v6)
```

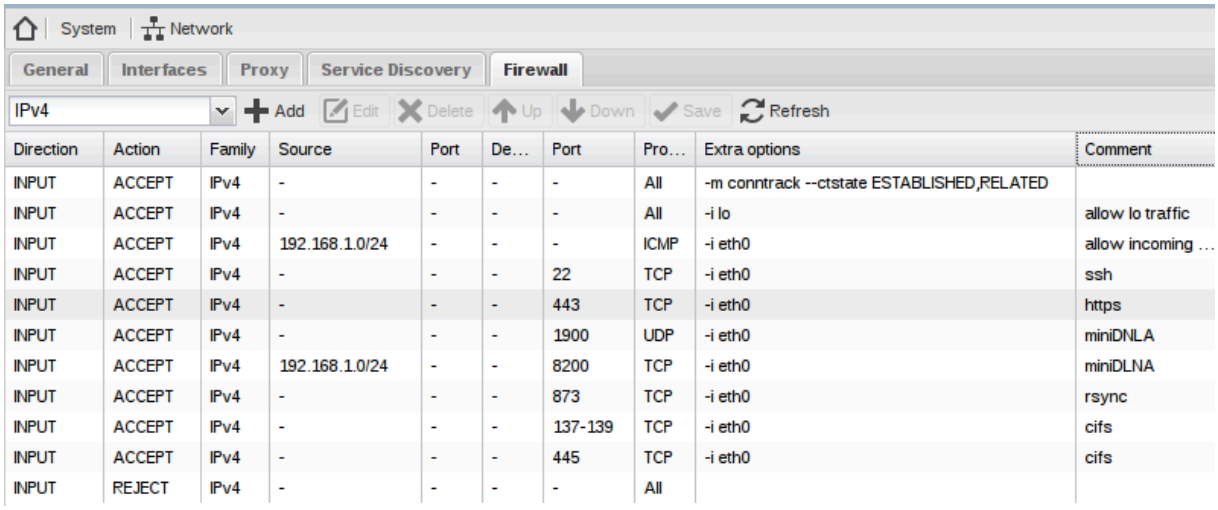
6.2. Firewall configuration with OMV

Alternatively you can configure your firewall rule by rule in *OpenMediaVault*:

```
OpenMediaVault -> System -> Network -> Firewall
```

A good template including other services can be found in this forum[15].

Below you find my sample settings for IPv4.



Direction	Action	Family	Source	Port	De…	Port	Pro…	Extra options	Comment
INPUT	ACCEPT	IPv4	-	-	-	-	All	-m conntrack --ctstate ESTABLISHED,RELATED	
INPUT	ACCEPT	IPv4	-	-	-	-	All	-i lo	allow lo traffic
INPUT	ACCEPT	IPv4	192.168.1.0/24	-	-	-	ICMP	-i eth0	allow incoming …
INPUT	ACCEPT	IPv4	-	-	-	22	TCP	-i eth0	ssh
INPUT	ACCEPT	IPv4	-	-	-	443	TCP	-i eth0	https
INPUT	ACCEPT	IPv4	-	-	-	1900	UDP	-i eth0	miniDNLA
INPUT	ACCEPT	IPv4	192.168.1.0/24	-	-	8200	TCP	-i eth0	miniDLNA
INPUT	ACCEPT	IPv4	-	-	-	873	TCP	-i eth0	rsync
INPUT	ACCEPT	IPv4	-	-	-	137-139	TCP	-i eth0	cifs
INPUT	ACCEPT	IPv4	-	-	-	445	TCP	-i eth0	cifs
INPUT	REJECT	IPv4	-	-	-	-	All		

Add similar rules for IPv6 or add a **REJECT from everywhere** rule to disable IPv6.

[15] http://forums.openmediavault.org/index.php/Thread/6411-Help-setting-up-firewall-iptables/

7. Install a DNLA/UPnP media server on your NAS (optional)

OpenMediaVault has management modules for many common protocols e.g. FTP, NFS, RSync, SMB/CIFS, SNMP, SSH and TFTP.

In order to set up a media server you may want to install the the *minidnla*-plugin

```
> apt-get install openmediavault-minidlna
```

Do not change the default settings, just define some shares on the tab

```
OpenMediaVault -> Services -> DNLA -> Shares
```

and enable the server:

```
OpenMediaVault -> Services -> DNLA -> Settings -> Enable
```

8. Set up a video/music player client (optional)

8.1. Windows 10 and Android

The Windows 10 Media Player and Yaacc application for Android can connect without any further configuration.

8.2. Debian

I recommend the *vlc* media-player.

```
vlc -> local network -> Universal Plug'n'Play
```

It may take a minute until *MiniDNLA* appears in the main window. Please be patient.

Firewall configuration

A simple firewall configuration tool is `gufw`.

```
> apt-get install gufw
```

The following could be a good start:

To	Action	From
8200/tcp	ALLOW	Anywhere
1900/udp	ALLOW	Anywhere
8200/tcp	ALLOW	Anywhere (v6)
1900/udp	ALLOW	Anywhere (v6)

9. References

- Migration Raid 1 to Raid 5 howto[16]
- Prepare disks für Raid[17]

[16] https://wiki.archlinux.org/index.php/Convert_a_single_drive_system_to_RAID
[17] https://wiki.archlinux.org/index.php/RAID