**OpenmediaVault and NextCloud on raspberryPI 3 installation steps**

**GENERAL:**

General idea is to have common home NAS to store own media and document files form different devices. This is not an fast and easy setup, so prepare yourself to dedicate some time to that project and make the things right from the beginning.

**Pre-requirements:**

* Wifi router DHCP enabled for LAN and WIFI
* Raspberry PI 3
* LAN connectivity between raspberry and your router or HDMI receiver (TV) to display raspberry console.
* USB disks with own power supply (you can power them up with additional USB hub) needed because raspberry cannot stand consumption load.
* PC

**Architecture:**

Most probably there are many ways to achieve described general capabilities. In this document, I will try to explain how to create your own home NAS box and interconnect it with your mobile phone and PC using raspberry pi 3 - as hosting equipment; openmediavault server - as core capabilities manager; nextCloud server and nextcloud mobile application for mobile/user data synchronization and shearing.

**SETUP:**

* **Raspberry pi 3**

We should first start with Raspberry PI 3 setup. As mentioned above we’ll use it to host our Openmediavault server. Openmediavault community ensured easy installation by providing special image for raspberry pi 2 and 3.

Openmediavault image can be downloaded from here:  
<https://sourceforge.net/projects/openmediavault/files/Raspberry%20Pi%20images/>

At the time of this document, recent version is - OMV\_3\_0\_79\_RaspberryPi\_2\_3\_4.9.35.7z from 07.16.2017

As a next step, we need to write the image to the sd card we dedicated for our raspberry. There are several good suggestions on how this could be done here:

<https://www.raspberrypi.org/documentation/installation/installing-images/>

Once the image is uploaded to raspberry we can continue with openmediavault setup and specifics.

* + **Setup wifi**

In case we deal with raspberry pi 3 – then we have a built-in wifi capabilities. That give us the opportunity to make wired LAN independent setup for our NAS. Nevertheless to setup the wifi we need LAN cable or HDMI receiver to display raspberry pi console.

Once we see the console output we can login with

user: admin

password: openmediavault

Openmediavault image is preconfigured with wired interface support (eth0) only, by default.

To enable our wifi module we should do following:

from: <http://www.linuxx.eu/2014/06/configure-wpa2-on-raspberry-pi.html>

Create config file with:

sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf

*ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev*

*update\_config=1*

*network={*

*ssid="SSID"*

*proto=RSN*

*key\_mgmt=WPA-PSK*

*pairwise=CCMP TKIP*

*group=CCMP TKIP*

*psk="yourkey"*

*}*

Edit interfaces file: (below lines should be uncommented)

sudo nano /etc/network/interfaces

allow-hotplug wlan0

iface wlan0 inet dhcp

wpa-conf /etc/wpa\_supplicant/wpa\_supplicant.conf

to ensure that we run most recent version of all OS level packages we should:  
sudo apt-get update

sudo apt-get upgrade

reboot

At this point we should be able to connect to the IP address that (in my case) DHCP server distributed for my device.

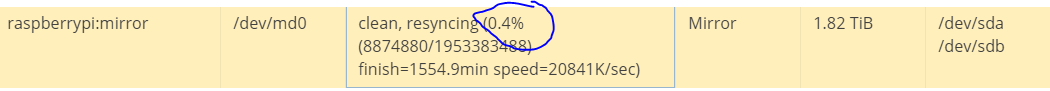
* **Home WIFI router setup**
* **Disks**

We should ensure that disks are connected and raspberry can see them. In my case command fdisk -l shows following:  
Disk /dev/**sda**: 1.8 TiB, 2000398929920 bytes, 488378645 sectors

Disk /dev/**sdb**: 1.8 TiB, 2000398929920 bytes, 488378645 sectors

Disks must stay clean - means without filesystem and in unmounted state so we can tie them in so called md device (multidevice driver, or software RAID) using openmediavault. In case openmediavault RAID manager cannot see the devices, ensure they are not mounted and wipe them using openmediavault wipe button under storage->physical disks.

**Attention: (read carefully below paragraph)**

Once Disks are available in RAID context menu, you can create wanted RAID array. In my case – RAID 1 (mirror). Once the RAID is visible in the UI we must wait MD devide initialization.  


This should become 100%. It takes a **lot** **of time** for resync to finish. (around **~24hours** for 2T USB 2.0 disks). Basically, this can be avoided with special property passed to md command line interface (--assume-clean). I haven’t found a clean way to inject such property to array management of openmediavault. In case you don’t feel comfortable with creating md device on OS level your own – take some rest and continue tomorrow☺

Once we are done with RAID management we can continue with filesystem creation.  
Filesystem creation as well takes some time (~30 minutes)! I picked ext4 as native Linux filesystem. (note: if you decide to kill the RAID array at some point in future and plug your HDD to windows, it wont recognize it’s filesystem. To be prepared for such case you better use NTFS which is native to windows OS and still manageable on Linux - ntfs-3g is the package that can work with that type of filesystem).

* + **Create shared folders and set permissions**

Once we are done with device initialization and our file system is in place we should continue with folder structures and permission sets.

Before that we should click on mount button at filesystem interface to mount our new device.

According to our goal we should consider having at least one folder where we manage our mobile data and documents and another one to manage RAW stuff. I personally decided to put all raspberry images that I have on such RAW-folder.

So I created following:  
Folder: nextCloud

Folder: RAWdata

Then I created dedicated application folder under nextCloud called app (nextCloud/app)

* **Install Nginx plugin**

Once we have initial folder structure in place we can start thinking about application installation and runtime. Since we decided to manage our mobile user data with NextCloud solution we can start preparing its https interface. There are many web servers that can serve NextCloud user interface. I decided to use Nginx since its very light weight and reach from capabilities perspective at the same time. Together with that there is a very handy plugin for OpenMediaVault so we can use Nginx out of the box with OpenMedia vault.  
Installation from menu: system -> Plugins -> openmediavault-nginx (current version 3.0.11)

Then select, click install, click apply if needed and in couple of seconds you’ll see additional service under services menu of OpenMediaVault.

That’s it.

* **Mysql setup**

Next cloud installation guide proposes to use mysql database instead of embedded sqlLite in case we would like better performance for bigger installations… At first glance it looks that overall application is not performing quite well from desktop computer so I decided to give MySQL a try.   
In Openmediavault there is already MySQL plugin that can be installed from plugins manager in the same way we installed Nginx. Once installed and visible in OpenMediaVault services section we can enable mysql admin site and configure a user for NextCloud application. Later we can refer to that account during NextCloud setup.

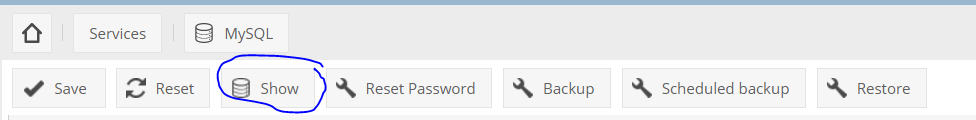
From mysql service page:

Enable networking – on

Bind address – 127.0.0.1

SQL management site – on

As soon as SQL management site is on you will be able to open management UI on https://<your IP>/mysql/ or by clicking at show button:



You can reset your password or look for default one to login and manage MySQL user list.

* **Configure Nginx Plugin**

Now we are well prepared for continue with NextCloud setup.

Before we move on to NextCloud installation we should prepare our web server to support NextCloud installation wizard.

Take some time to read complete document form NextCloud admin guide here:

<https://docs.nextcloud.com/server/12/admin_manual/installation/nginx.html>

before continue with server setup below.

Back to OpenMediVault. We should create certificate under system -> certificates as we’ll use it later to setup https server.

Before we setup the server we should ensure that we have a PHP-PFM pool to associate with. Under pools tab click on add.

User – www-data

Group – www-data

Then under services -> Nginx setup an Nginx server.

From tab settings, we first enable the server.

From servers tab we click on add button to create so called server (site configuration for Nginx)

In server configuration dialog:

choose app directory that we created earlier (nextCloud/app).

Use root – on

Enable SSL – on

Default server – on

Certificate – pick the one we already have.

Only SSL – on

Enable PHP – on

PHP-PFM pool – the one we created earlier

Use Index – on

Index.html – on

Index.php – on

Enable log – on

Extra options – copy and paste configuration suggested in NextCloud admin guide we read thru earlier.

I did the “Nextcloud in the webroot of nginx” extra configuration.

* **Install NextCloud**

NextCloud put some efforts to make installation and configuration easy and quick. They made web installer application for us.

Copy web installer in your nextCloud/app folder. Here is how and from where:  
<https://hagen.cocoate.com/2016/08/21/nextcloud-installation-shared-hosting-first-steps/>

I cloned web installer form here:

git clone <https://github.com/nextcloud/web-installer.git>

* **System -> Scheduled jobs**

This is very handy feature. I personally included md device scanning to automatically recognize my md device in case of reboot:  
time - at reboot

User - root

Command - mdadm --assemble --scan --verbose

* **Problems that I faced during installation**

Socket error:

Default socket set does not work for some reason an nginx configuration line fastcgi\_pass unix:/var/run/php5-fpm.sock; solves that problem.

To debug: There must be a configuration under /etc/php5/fpm/pool.d. The name corresponds to unique UUID that is reposted as not found. For example:

connect() to unix:/var/run/fpm-c946f172-7e5b-40da-8158-db8f68fcb024.sock failed (2: No such file or directory)

in case configuration is not in place we should try to create another pool and associate it with nginx server. Or we can point the server to default socket using - fastcgi\_pass unix:/var/run/php5-fpm.sock instead of fastcgi\_pass $socket; in extra options we pass to nginx server configuration.