ownCloud 8 Server auf RaspberryPi (Basis: Raspbian Wheezy)

Stand: 13.05.2015 Quelle(n): http://www.forum-raspberrypi.de/, Community FSOG (Eugen Albiker, Edgar „Fast Edi“ Hoffmann, Jannis Pinter)

Inhalt

In dieser Anleitung erklären wir, wie mittels der Freien Software ownCloud aus einem Raspberry Pi ein Cloud Storage Server à la Dropbox gemacht wird.

Damit die ownCloud auf dem RaspberryPi in angenehmer Geschwindigkeit läuft, werden die Softwarepakete „php-pear“ und „php-apc“ mit installiert.

Auf unseren Server installieren wir zusätzlich noch:

- Apache2 oder nginx Webserver
- PHP5
- SQLite (Datenbank auf Dateibasis)

Grundlage ist das Betriebssystem Raspbian 2015-05-05-raspbian bzw. das Gesamtpaket NOOBS 1.4.1 (Stand jeweils 13.05.2015), welche von der Internetseite des Projektes heruntergeladen werden können.

Vorarbeiten

Installation des Grundsystems

Download der aktuellen Version von Debian Wheezy bzw. NOOBS für den Raspberry Pi unter www.raspberrypi.org/downloads

Unter Linux ist die Installation des Pi-Images nicht sehr aufwendig.

Eine Erklärung, wie man das Debian Wheezy Image auf die SD-Karte bekommt, findet sich unter: www.elinux.org/RPi_Easy_SD_Card_Setup

Standard-Login: Benutzer „pi“ , Passwort „raspberry“

Wenn der frisch installierte Pi startet, sollte man in einem Menü landen: „raspi-config“.

Sollte das nicht passieren, startet man dieses von Hand:

```
sudo raspi-config
```

Hier sollte man die root Partition an die Größe der SD-Karte angleichen (bei Verwendung von Raspbian) und den Raspberry Pi übertakten

- *Expand Filesystem* ausführen damit wir genug Platz für Pakete, ownCloud an sich und Uploads haben (bei Verwendung von Raspbian)
- Unter Advanced Options *Memory Split* auf „16“ MB einstellen. Dies ist die kleinstmögliche Einstellung. Die GPU bekommt somit 16MB.
- overclock auf „Medium“
- SSH-Verbindung aktivieren, falls man von Aussen auf den Pi zugriefen möchte (empfehlenswert)
- „Finish“ und danach die Frage nach dem Reboot mit „Yes“ beantworten.

Paketlisten und System aktualisieren

```
sudo apt-get update
sudo apt-get upgrade
```

Variante A: Mit Apache

Benötigte Pakete installieren

```
sudo apt-get install apache2 php5 php5-gd php5-sqlite php5-curl php5-common php5-intl php-pear php-apc php-xml-parser libapache2-mod-php5 curl libcurl3 libcurl3-dev sqlite
```

Apache Host

Damit Apache beim Start nicht meckert, tragen wir am Ende der Datei „apache2.conf“ noch den Hostnamen des Webservers ein:

```
sudo nano /etc/apache2/apache2.conf
```

ServerName ownCloud

Hier muss auch noch der Hostname eintragen werden:

```
sudo nano /etc/hosts
```

127.0.1.1 raspberrypi ownCloud

ownCloud Rechte gewähren

```
sudo nano /etc/apache2/sites-enabled/000-default
```

In folgendem Abschnitt (und nur da!) „AllowOverride None“ zu „AllowOverride All“ ändern

```
<Directory /var/www/>
      Options Indexes FollowSymLinks MultiViews
      AllowOverride All
      Order allow,deny
      allow from all
</Directory>
```

PHP für Apache anpassen/konfigurieren

```
sudo nano /etc/php5/apache2/php.ini
```

Hier passen wir „upload_max_filesize“ und „post_max_size“ an.

Damit gehen wir sicher, daß PHP auch größere Dateien hochläd und nicht abbricht! Wir wählen hier „2G“.

PHP gestattet die Angabe von K für Kilo, M für Mega und G für Giga bei der Angabe von Werten und rechnet diese automatisch um.

```
upload_max_filesize = 2G
post_max_size = 2G
```

Kleiner Tipp: Die php.ini ist eine sehr lange Datei. Es bietet sich also an die beiden Begriffe („upload_max_filesize“ sowie „post_max_size“) mit „STRG + W“ zu suchen. Damit ersparst du dir unnötige Scroll-Zeit.

Apache anpassen

Die Datei „index.html“ aus dem „/var/www/“ Verzeichnis löschen:

```
sudo rm /var/www/index.html
```

Danach aktivieren wir noch die Apache Rewrite Engine mit folgenden Befehlen:

```
sudo a2enmod rewrite
sudo a2enmod headers
```

Nun starten wir Apache neu...

```
sudo /etc/init.d/apache2 restart
```

...und können mit Kapitel 4, der Installation der ownCloud fortfahren.

Variante B: Mit nginx

Benötigte Pakete installieren

```
sudo apt-get install nginx openssl ssl-cert php5-cli php5-sqlite php5-gd php5-curl php5-common php5-cgi sqlite php-pear php-apc libapr1 libtool curl libcurl4-openssl-dev php-xml-parser php5 php5-dev php5-fpm memcached php5-memcache varnish
```

SSL Zertifikat erstellen (gültig für 1 Jahr)

```
sudo openssl req $@ -new -x509 -days 365 -nodes -out /etc/nginx/cert.pem -keyout /etc/nginx/cert.key
sudo chmod 600 /etc/nginx/cert.pem
sudo chmod 600 /etc/nginx/cert.key
```

Werte anpassen, um den RaspberryPi nicht zu überlasten

Nun passen wir noch ein paar Werte bezüglich der maximalen Prozesse an, sodass Nginx den Raspberry Pi nicht überlastet.

```
sudo sed -i "s/worker_processes 4;/worker_processes 1;/g" /etc/nginx/nginx.conf
sudo sed -i "s/worker_connections 768;/worker_connections 128;/g" /etc/nginx/nginx.conf
```

Danach kann Nginx gestartet werden

```
sudo /etc/init.d/nginx start
```

nginx Webserver konfigurieren

```
sudo nano /etc/nginx/sites-available/default
```

Hier löscht man den kompletten Inhalt und fügt stattdessen den folgenden ein.

Darauf achten, dass die IP-Adresse „192.168.x.xxx“ mit der des eigenen Raspberry Pi ersetzt wird. Diese findet man mit dem Befehl „*ifconfig*“. Je nachdem wie der Pi verbunden ist, kann man den Devices “eth” oder “wlan” die IP-Adresse aus folgender Zeile entnehmen: „inet addr:192.168.x.xxx Bcast:192.168.0.255 Mask:255.255.255.0“.

```
server {
      listen 80;
      server_name 192.168.x.xxx;
      return 301 https://$host$request_uri;
}
server {
      listen 443 ssl;
      server_name 192.168.x.xxx;
      ssl_certificate /etc/nginx/cert.pem;
      ssl_certificate_key /etc/nginx/cert.key;
      root /var/www/owncloud;
      index index.php;
      client_max_body_size 2G; # set maximum upload size
      fastcgi_buffers 64 4K;
      location ~ ^/(data|config|\.ht|db_structure\.xml|README) {
      deny all;
}
location / {
      try_files $uri $uri/ index.php;
}
location @webdav {
      fastcgi_split_path_info ^(.+\.php)(/.*)$;
      fastcgi_pass 127.0.0.1:9000;
      fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
      fastcgi_param HTTPS on;
      include fastcgi_params;
}
location ~ ^(?<script_name>.+?\.php)(?<path_info>/.*)?$ {
      try_files $script_name = 404;
      include fastcgi_params;
      fastcgi_param PATH_INFO $path_info;
      fastcgi_param HTTPS on;
      fastcgi_pass 127.0.0.1:9000;
}
}
```

PHP für nginx anpassen/konfigurieren (Schritt 1)

```
sudo nano /etc/php5/fpm/php.ini
```

Hier passen wir „upload_max_filesize“ und „post_max_size“ an.

Damit gehen wir sicher, daß PHP auch größere Dateien hochlädt und nicht abbricht!

Wir wählen hier „2G“.

PHP gestattet die Angabe von K für Kilo, M für Mega und G für Giga bei der Angabe von Werten und rechnet diese automatisch um.

```
upload_max_filesize = 2G
post_max_size = 2G
```

Kleiner Tipp: Die php.ini ist eine sehr lange Datei. Es bietet sich also an die beiden Begriffe („upload_max_filesize“ sowie „post_max_size“) mit „STRG + W“ zu suchen. Damit erspart man sich unnötige Scroll-Zeit.

An das Ende der Datei wird nun noch folgendes für die Aktivierung des Caching und den Speicherort der temporären Dateien angefügt:

```
upload_tmp_dir = /srv/http/owncloud/data
extension = apc.so
apc.enabled = 1
apc.include_once_override = 0
apc.shm_size = 256
```

Als nächstes wird dieser Ordner mit den dazugehörigen Rechten erstellt:

```
sudo mkdir -p /srv/http/owncloud/data
sudo chown www-data:www-data /srv/http/owncloud/data
```

PHP für nginx anpassen/konfigurieren (Schritt 2)

```
sudo nano /etc/php5/fpm/pool.d/www.conf
```

Hier wird die folgende Zeile von:

```
listen = /var/run/php5-fpm.sock
```

in

```
listen = 127.0.0.1:9000
```

geändert.

Nun den Webserver und PHP neu starten:

```
sudo /etc/init.d/php5-fpm restart
sudo /etc/init.d/nginx restart
```

Download und Installation von ownCloud

Es gibt mehrere Möglichkeiten, die ownCloud zu installieren.

Entweder manuell, d.h. man besorgt sich das Installationspaket und richtet ownCloud von Hand ein, oder aber über ein bereitgestelltes Repository. Hier kann dann wie gewohnt über den Befehl *apt-get install* verfahren werden.

Die Installation über das Repository ist empfehlenswert, da so auch immer Updates verfügbar sind.

Manuell

Aktuelle Version beachten! In dieser Anleitung wurde die Version 8.0.0 benutzt (Stand: 17.02.15).

```
cd
wget http://download.owncloud.org/community/owncloud-8.0.0.tar.bz2
```

Nun wird das heruntergeladene Archiv entpackt:

```
tar xvf owncloud-8.0.0.tar.bz2
```

Dann wird ownCloud aus dem Download Verzeichnis in das Webverzeichnis verschoben und dem User „www-data“ noch die benötigten Rechte erteilt:

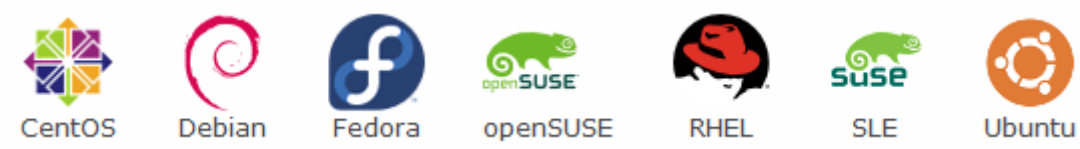
```
sudo mv owncloud /var/www/
sudo chown -R www-data:www-data /var/www/owncloud
```

Zuletzt kann noch der Download gelöscht werden und die Installation ist soweit fertig.

```
rm -rf owncloud owncloud-8.0.0.tar.bz2
```

Mittels openSUSE Repository

Das von der openSUSE Community bereitgestellte Repository (http://software.opensuse.org/download/package?project=isv:ownCloud:community&package=owncloud) steht für diverse Linux-Varianten zur Verfügung.



Für die in dieser Anleitung verwendete Distribution Raspbian/Debian sind folgende Schritte notwendig:

Zunächst muss das Repository eingebunden werden

```
sudo su
# Debian 8:
echo 'deb http://download.opensuse.org/repositories/isv:/ownCloud:/community/Debian_8.0/ /'
>> /etc/apt/sources.list.d/owncloud.list
# Debian 7:
echo 'deb http://download.opensuse.org/repositories/isv:/ownCloud:/community/Debian_7.0/ /'
>> /etc/apt/sources.list.d/owncloud.list
```

Wichtig: Bei den beiden Anführungszeichen (hier in rot dargestellt) handelt es sich um das Apostroph!

Danach sollte man (immer noch als root) noch den Vertrauensschlüssel des Repositories importieren, da es sonst zu Fehlermeldungen kommt. Bei Fremdquellen ist allerdings immer Vorsicht geboten, da man diesen durch den Schlüsselimport vertraut.

```
# Debian 8:
wget
http://download.opensuse.org/repositories/isv:ownCloud:community/Debian_8.0/Release.key
apt-key add - < Release.key
# Debian 7:
wget
http://download.opensuse.org/repositories/isv:ownCloud:community/Debian_7.0/Release.key
apt-key add - < Release.key
```

Ist dies erledigt, kann man die Paketliste aktualisieren und ownCloud installieren.

```
apt-get update
apt-get install owncloud
```

Owncloud-Dateien auf USB Stick / Externe Festplatte (optional)

Es ist auch möglich, eine externe Festplatte am Raspberry Pi anzuschließen und dort die ownCloud-Dateien zu installieren. Wenn dies gewünscht ist, sind nach dem Kapitel „Download und Installation von ownCloud“ die folgenden Schritte zu befolgen.

Schritt 9 entfällt, bzw. ist bei dem optionalen Schritt noch etwas anders.

USB Stick / Externe Festplatte anschließen

Nachdem die Festplatte angeschlossen hast, müssen wir herausfinden, wie diese heißt, formatieren und mounten.

Name herausfinden

```
sudo fdisk -l
```

Die Ausgaben könnte folgende sein:

```
Device       Boot   Start  End          Blocks       Id     System
/dev/sda1           32     7829503      3914736      b      W95 FAT32
```

Mein USB-Stick ist also als „sda1“ bekannt. Den Namen merken wir uns und fahren fort.

Formatieren und mounten

```
sudo mkfs.ext4 /dev/sda1
sudo mkdir -p /media/usb1/ownCloud/data
sudo mount /dev/sda1 /media/usb1
```

Nun müssen wir noch festlegen, daß nach jedem Reboot der mount automatisch ausgeführt wird.

Dazu editieren wir die Datei „/etc/fstab“ und fügen in der letzten Zeile unsere „sda1“ ein:

```
sudo nano /etc/fstab
```

So sollte es dann aussehen:

```
proc              /proc        proc   defaults           0     0
/dev/mmcblk0p1    /boot        vfat   defaults           0     2
/dev/mmcblk0p2    /            ext4   defaults,noatime   0     1
/dev/sda1         /media/usb1  ext4   defaults           0     0
```

Rechte vergeben

```
sudo chown -R www-data:www-data /var/www
sudo chown -R www-data:www-data /media/usb1/ownCloud/data
```

Verzeichnis in Owncloud-Konfiguration hinterlegen

Nun kann man unter der IP des Raspberry Pi den ownCloud-Server unter (http://192.168.x.xxx/owncloud) aufrufen.

Man muss nun noch einen ownCloud-User anlegen und das neue Verzeichnis einstellen. Dazu klickt man auf „Fortgeschritten“ und trägt in Datenverzeichnis folgendes ein: „/media/usb1/ownCloud/data“ Danach klickt man auf „Installation abschließen“.

Nun führen wir noch einen Neustart des Apache Servers aus.

```
sudo /etc/init.d/apache2 start
```

Fertig! Der Schritt „Konfiguration beenden“ entfällt!

Konfiguration beenden

Nun kann man unter der IP des RaspberryPi den ownCloud-Server (http://192.168.x.xxx/owncloud) aufrufen.

Wenn man den ownCloud Server auch außerhalb des eigenen LANs benutzen möchte und zum Beispiel mit dem Smartphone von unterwegs zugreifen will, hilft das Kapitel „ownCloud von Aussen ansprechen“ weiter hinten weiter.

Desktop-Client installieren

Für die automatische Synchronisation eines Verzeichnisses auf dem eigenen (Desktop-) Rechner verwendet man am Besten einen der verfügbaren Desktop-Clients.

Linux

Auf der Webseite des ownCloud Projektes stehen Installationspakete für die gängigen Distributionen (Debian, Fedora, openSUSE, Ubuntu,...) zum Download bereit. Praktischer ist es jedoch, die entsprechenden Repositories einzubinden, sodass immer die aktuellste Version verwendet werden kann.

```
# Debian 8:
sudo echo 'deb
://download.opensuse.org/repositories/isv:ownCloud:desktop/Debian_8.0/Release/ /' >>
/etc/apt/sources.list.d/owncloud-client.list
# Debian 7:
sudo echo 'deb
://download.opensuse.org/repositories/isv:ownCloud:desktop/Debian_7.0/Release/ /' >>
/etc/apt/sources.list.d/owncloud-client.list
```

Wichtig: Bei den beiden Anführungszeichen (hier in rot dargestellt) handelt es sich um das Apostroph!

Danach sollte man, falls noch nicht wie in Abschnitt 4.2 geschehen, (immer noch als root) den Vertrauensschlüssel des Repositories importieren, da es sonst zu Fehlermeldungen kommt. Bei Fremdquellen ist allerdings immer Vorsicht geboten, da man diesen durch den Schlüsselimport vertraut.

```
# Debian 8:
sudo wget
http://download.opensuse.org/repositories/isv:ownCloud:desktop/Debian_8.0/Release.key
sudo apt-key add - < Release.key
# Debian 7:
sudo wget
http://download.opensuse.org/repositories/isv:ownCloud:desktop/Debian_7.0/Release.key
sudo apt-key add - < Release.key
```

Hier nicht den Bindestrich im letzten Befehl vergessen!

Ist dies erledigt, kann man die Paketliste aktualisieren und ownCloud installieren.

```
sudo apt-get update
sudo apt-get install owncloud-client
```

Windows und Mac

Die aktuellen Windows- bzw. Mac-Clients kann man als Installationspakete direkt herunterladen und installieren.

SSL bzw HTTPS mit einer CA und selbst signiertem Zertifikat hinzufügen (optional)

Apache

```
sudo mkdir -p /etc/apache2/ssl
```

Privaten Schlüssel für die CA generieren (Durch Passphrase geschützt):

```
sudo openssl genrsa -aes256 -out /etc/apache2/ssl/CA.key 4096
```

Öffentliches Zertifikat für die CA erzeugen:

```
sudo openssl req -x509 -new -nodes -key /etc/apache2/ssl/CA.key -days 1095 -sha512 -out
/etc/apache2/ssl/CA.pem
```

Privaten Schlüssel und CSR für die ownCloud-Installation generieren:

```
sudo openssl req -new -newkey rsa:4096 -nodes -sha512 -out /etc/apache2/ssl/apache.csr
-keyout /etc/apache2/ssl/apache.key
```

OwnCloud-Zertifikat durch die CA signieren:

```
sudo openssl x509 -req -in /etc/apache2/ssl/apache.csr -CA /etc/apache2/ssl/CA.pem
-CAkey /etc/apache2/ssl/CA.key
      -CAcreateserial -out /etc/apache2/ssl/apache.pem -days 1095
```

Schlüssel und Zertifikat in Eins packen:

```
sudo cat /etc/apache2/ssl/apache.key /etc/apache2/ssl/CA.pem >> /etc/apache2/ssl/apache.pem
```

Die überflüssigen Dateien löschen:

```
sudo rm /etc/apache2/ssl/apache.csr
sudo rm /etc/apache2/ssl/apache.key
```

Zuletzt noch sichere Berechtigungen setzen:

```
sudo chmod 400 /etc/apache2/ssl/CA.key
sudo chmod 444 /etc/apache2/ssl/CA.pem
sudo chmod 400 /etc/apache2/ssl/apache.pem
```

Apache muss auf TCP Port 443 lauschen.

Hierzu ändert man die Datei so, daß alle „#“ verschwinden, die dort nicht hingehören.

```
sudo nano /etc/apache2/ports.conf
```

```
Listen 192.168.xxx.xxx:80
<IfModule mod_ssl.c>
      Listen 443
</IfModule>
```

Apache reloaden:

```
sudo service apache2 reload
```

SSL Modul aktivieren sowie einen force-reload von Apache2 durchführen:

```
sudo a2enmod ssl
sudo service apache2 force-reload
```

SSL Webseite konfigurieren (vHost):

```
sudo nano /etc/apache2/sites-available/ssl
```

```
<virtualhost *:443>
      SSLEngine On
      SSLCertificateFile /etc/apache2/ssl/apache.pem
      DocumentRoot /var/www
</virtualhost>
```

Die neue vHost Konfiguration aktivieren sowie einen force-reload von Apache2 durchführen:

```
sudo a2ensite ssl
sudo service apache2 force-reload
```

Nun sollte die ownCloud unter https://192.168.xxx.xxx erreichbar sein.

Andere Vorgehensweise

Die Zertifikate nutzen den modernen SHA512-Hash, sind 4096 Bit lang und der private Schlüssel der CA wird AES verschlüsselt gespeichert. Die Zertifikate sind 3 Jahre gültig, danach müssen Sie erneuert werden.

Privaten Schlüssel für die CA generieren (Durch Passphrase geschützt):

```
openssl genrsa -aes256 -out CA.key 4096
```

Öffentliches Zertifikat für die CA erzeugen:

```
openssl req -x509 -new -nodes -key CA.key -days 1095 -sha512 -out CA.pem
```

Privaten Schlüssel und CSR für die OwnCloud-Installation generieren:

```
openssl req -new -newkey rsa:4096 -nodes -sha512 -out
/etc/ssl/certs/username.dyndns.org.csr -keyout
      /etc/ssl/private/username.dyndns.org.key
```

OwnCloud-Zertifikat durch die CA signieren:

```
openssl x509 -req -in /etc/ssl/certs/username.dyndns.org.csr -CA CA.pem -CAkey CA.key
-CAcreateserial -out      /etc/ssl/certs/username.dyndns.org.crt -days 1095
```

Den CSR kann man nun löschen:

```
rm /etc/ssl/certs/username.dyndns.org.csr
```

Zuletzt noch sichere Berechtigungen setzen:

```
chmod 400 CA.key
chmod 444 CA.pem
chmod 400 /etc/ssl/private/username.dyndns.org.key
chmod 444 /etc/ssl/certs/username.dyndns.org.crt
```

Zertifikat in Android importieren

Mit dem Befehl

```
openssl x509 -in /etc/apache2/ssl/apache.pem -outform DER -out /etc/apache2/ssl/apache.crt
```

Kann das Zertifikat in ein Android-konformes Format gebracht werden, um es unter Einstellungen, Sicherheit, Von SD-Karte installieren zu installieren.

Cipher-Suite anpassen

Der Client (Webbrowser, App, etc.) nutzt immer die Cipher-Suite die vom Server vorgegeben wird und verwendet, falls verfügbar, die am höchsten priorisierte.

Im Apache Webserver sollte man deshalb am Besten die Cipher-Suite anpassen.

Dies geschieht in der VirtualHost-Sektion der OwnCloud-Installation:

```
SSLProtocol all -SSLv2 -SSLv3
SSLCompression Off
SSLHonorCipherOrder on
SSLCipherSuite "ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES128-GCM-SHA256 DHE-RSA-AES256-
SHA256 DHE-RSA-AES256-SHA
      !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS !RC4"
```

(Hinweis: Der letzte Eintrag ist eine Zeile!)

Damit ist das ganze nicht nur sauber verschlüsselt, sondern bietet auch noch Perfect-Forward-Secrecy, was die zurückliegende Kommunikation auch dann schützt, wenn mal der geheime Schlüssel abhanden kommt.

Schlüsselaustausch-Verfahren anpassen

Diffie-Hellman ist ein sehr sicheres Schlüsselaustausch-Verfahren. Dabei werden Sitzungsschlüssel ausgetauscht, welche genauso lang sind, wie die Diffie-Hellman Parameter. Diese temporär erzeugten Schlüssel werden dann zur Verschlüsselung genutzt.

Apache 2.2 kann aber nur Diffie-Helmman bis 1024 Bit (entspricht einem 1024 Bit langem RSA Key).

Debian liefert in der stable Version aber keine neuere Version (Stand: 15.04.14).

Deshalb könnte man auf Apache 2.4.9 aus den Testing-Quellen umsteigen.

Vor einer solchen Aktion jedoch als Vorsichtsmaßnahme am besten die SD-Karte des RaspberryPi sichern, damit man gegebenenfalls alles wieder zurückspielen kann.

Nach der Sicherung dann in der Datei */etc/apt/sources.list* folgende Zeilen einfügen:

```
deb http://ftp.debian.org/debian/ jessie main
deb-src http://ftp.debian.org/debian/ jessie main
deb http://security.debian.org/ jessie/updates main
deb-src http://security.debian.org/ jessie/updates main
```

Danach in der Datei */etc/apt/apt.conf* folgende Zeile einfügen:

```
APT::Default-Release "stable";
```

Dann nochmal

```
apt-get update && apt-get upgrade
```

damit alles auf dem aktuellen Stand ist.

Nun Apache 2.4.9 aus den Testing-Quellen beziehen:

```
apt-get -t testing install apache2
```

Das dürfte ein paar Pakete aktualisieren, da der Apache 2.4.9 aktuellere Abhängigkeiten hat.

Wenn alles geklappt hat, kurzerhand mal testen:

```
service apache2 restart
```

Dann kann man sich die 4096 Bit DH-Parameter generieren:

```
openssl dhparam 4096 > /etc/ssl/dh4096.pem
```

Und in der */etc/apache2/mods-available/ssl.conf* folgendes hinzufügen:

```
SSLDHParametersFile /etc/ssl/dh4096.pem
```

Apache nochmal neu starten, fertig!

Downgrade-Angriffe verhindern

Um die Sicherheit der ownCloud noch weiter zu verbessern, kann man HSTS (HTTP Strict Transport Security) aktivieren.

Das schützt gegen sogenannte Downgrade-Angriffe.

HSTS sorgt dafür, dass der Browser nur noch HTTPS-Verbindungen zur ownCloud aufbaut, sobald einmal eine HTTPS-Verbindung zustande kam.

Normalerweise ist zwar sowieso nur Port 443 auf die ownCloud weitergeleitet, aber jemand könnte die DNS-Auflösung manipulieren, und dich auf einen fremden Server weiterleiten.

Irgendwo im ownCloud VirtualHost Eintrag der */etc/apache2/sites-available/default-ssl.conf* (kann auch anders benannt sein) muss man noch folgendes einfügen:

```
Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"
```

Und danach mit

```
a2enmod Headers
```

das Apache Modul aktivieren und den Apache neu starten.

Weitere Tips, Tricks und Vorgehensweisen

OwnCloud „von Außen“ ansprechen

Wenn man die ownCloud nicht nur innerhalb des eigenen Netzwerkes nutzen möchte, sondern auch von Außen, also über das Internet, sind verschiedene Dinge zu beachten.

Zunächst benötigt man eine feste IP. Da jedoch private „Normalanwender“ dies auch aus Kostengründen wohl eher nicht haben dürften, bietet sich ein sogenannter dynamischer DNS Dienst an.

Einer davon ist z.B. DynDNS.

Hat man nun eine DynDNS-Adresse, muss diese noch im Router (z.B. Fritz!Box) eingetragen, sowie eine Portweiterleitung auf die ownCloud eingestellt werden.

Feste IP-Adresse statt DHCP

Feststellen der IP-Adresse des Gateways/Routers mittels

```
netstat -r -n
```

Danach muss die Datei */etc/network/interfaces* bearbeitet werden.

```
sudo nano /etc/network/interfaces
iface eth0 inet dhcp
ersetzen durch
iface eth0 inet static
      address 192.168.xxx.xxx
      netmask 255.255.255.0
      gateway 192.168.xxx.xxx
```

Zuletzt starten wir das Netzwerkdienstprogramm neu, damit die Einstellungen übernommen werden.

```
sudo /etc/init.d/networking restart
```

Den Pi über den Hostnamen ansprechen (statt über IP)

Wenn man den Pi auch über den vergebenen Hostnamen ansprechen möchte, muss man den avahi-daemon (zeroconf) verwenden. Dieser wird folgendermaßen installiert:

```
sudo apt-get install avahi-daemon
```

Danach den Autostart für den avahi-daemon einrichten:

```
sudo insserv avahi-daemon
```

Und die neue Konfiguration aktivieren:

```
sudo /etc/init.d/avahi-daemon restart
```

Der Pi sollte nun über den vergebenen Hostnamen (z.B. raspberrypi.local oder http://raspberrypi.local) ansprechbar sein. Wenn die Verbindung über SSH unter Windows nicht funktioniert, könnte das am fehlenden Bonjour-Service liegen. Dieser muss dann nachinstalliert werden (http://support.apple.com/kb/DL999).

Von USB booten anstatt von SD-Karte

Dazu muss man in der Datei „*cmdline.txt*“ den Eintrag:

```
disk=/dev/mmcblk0p2
```

ändern in

```
disk=/dev/sda
```

SWAP Space erhöhen

Zuerst melden wir und mittels folgendem Kommando als Root-User an. Dies ist vonnöten, um das SWAP File zu erzeugen.

```
sudo su
```

Nun erzeugen wir das SWAP-File. In diesem Fall setzte ich die Größe des SWAPs auf 512 MB.

Generell gilt die Faustregel: *RAM * 2 = SWAP*.

Dementsprechend sollten die diejenigen von euch, die ein Modell mit 512 MB haben den SWAP auf 1024 MB einstellen.

```
echo "CONF_SWAPSIZE=512" > /etc/dphys-swapfile
dphys-swapfile setup
```

Nachdem wir das SWAP-File erzeugt haben müssen wir dieses noch einmalig aktivieren und uns anschliessend noch als root abmelden.

```
dphys-swapfile swapon
exit
```

Fertig! Nun hat der RaspberryPi für den Fall der Fälle genug SWAP und wird wegen einer Überladung des RAMs nicht in die Knie gehen.

Keine SWAP-Datei anlegen

```
sudo apt-get purge dphys-swapfile
```

Anschließend „sudo reboot“ ausführen. Danach mit

```
free -h
```

überprüfen, daß 0B SWAP zur Verfügung steht.

Email konfigurieren I

```
sudo apt-get install exim4
sudo dpkg-reconfigure exim4-config
      →mail sent by smarthost; received via SMTP or fetchmail
      →System mail name: z.B. „www.meineOwnCloud.de“ (FQDN des DynDNS)
      →IP-adresses to listen: 127.0.0.1 ; ::1 (Vorschlag lassen)
      →Other destinations for which…: raspberrypi (Hostname des Pi, Vorschlag lassen)
      →Machines to relay mail for: leer lassen
      →IP adress or host name of the outgoing smarthost: z.B.: „smtp.1und1.de::587“(SMTP-
Host des Providers und Port)
      →Hide local mail name in outgoing mail: No
      →Keep number of DNS-queries minimal: No
      →Delivery merthod for local mail: „Maildir format in home directory“
      →Split configuration into small files: No
      →Root ans postmaster mail recipient: Leer lassen
```

In der Datei */etc/exim4/passwd.client* eintragen:

```
smtp.provider.de:<LoginDerMailadresse>:<passwort>
```

Exim4 Konfiguration updaten:

```
sudo update-exim4.conf
sudo /etc/init.d/exim4 restart
```

Test mit: echo 'ok' | mail -s '1und1 SMTP Relay Test' empfaender@provider.de

Email konfigurieren II

Zuerst die benötigten Pakete installieren:

```
sudo apt-get install ssmtp heirloom-mailx
sudo apt-get install mailutils
sudo apt-get install mpack
```

Dann in der Datei „/etc/ssmtp/ssmtp.conf“ die Standardeinstellungen für SSMTP festlegen:

```
AuthUser=youruserid@gmail.com
AuthPass=userpass
FromLineOverride=YES
mailhub=smtp.gmail.com:587
hostname=<hostname des raspberrypi>
UseSTARTTLS=YES
```

Um die Konfiguration zu testen kann nun folgendes eingegeben werden:

```
echo "Das ist ein Test" | mail -s "Betreff xyz" empfaenger@provider.de
```

Wenn Anhänge verschickt werden sollen, sollte das folgende funktionieren:

```
mpack -s "Das ist ein Test" /home/pi/test/datei.txt empfaenger@provider.de
```

ownCloud Thema anpassen

Um der ownCloud ein individuell angepasstes Aussehen zu geben, kann man das Thema anpassen.

Dies geschieht am einfachsten, indem man das Standardthema kopiert.

Im Minimalfall sind lediglich die Verzeichnisse „css“ und „img“ notwendig.

Abgelegt und bearbeitet wird die Kopie dann im Verzeichnis /themes/themenname.

Logo der Anmeldeseite ändern

Um das Logo der Anmeldeseite zu ändern, sollte man zwei Dateien anpassen:

- logo.svg (logo.png)
- logo-wide.svg (logo-wide.png)

Diese befinden sich im Verzeichnis /img

Farbverlauf im Kopfbereich der Anmeldeseite ändern

Auch die Farbe des oberen Balkens auf der Anmeldeseite kann individuellen Bedürfnissen angepasst werden.

Dazu muss man in der Datei styles.css, welche sich im vorher kopierten Verzeichnis /themes/themename/css befindet, die unterstrichenen, roten Hex-Farbwerte mit den gewünschten Start- bzw. Endfarbwerten überschreiben.

```
#body-login {
      text-align: center;
      background: #1d2d44; /* Old browsers */
      background: url('../img/noise.png'), -moz-linear-gradient(top, #35537a 0%, #1d2d44
100%); /* FF3.6+ */
      background: url('../img/noise.png'), -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#35537a),
            color-stop(100%,#1d2d44)); /* Chrome,Safari4+ */
      background: url('../img/noise.png'), -webkit-linear-gradient(top, #35537a 0%,#1d2d44
100%); /* Chrome10+,Safari5.1+ */
      background: url('../img/noise.png'), -o-linear-gradient(top, #35537a 0%,#1d2d44
100%); /* Opera11.10+ */
      background: url('../img/noise.png'), -ms-linear-gradient(top, #35537a 0%,#1d2d44
100%); /* IE10+ */
      background: url('../img/noise.png'), linear-gradient(top, #35537a 0%,#1d2d44
100%); /* W3C */
      filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#35537a',
endColorstr='#1d2d44',GradientType=0 ); /* IE6-9 */
        }
```

Um das Thema auch zu laden, muss man dieses in der Datei /var/www/owncloud/config/config.php noch angeben:

```
‘theme‘ => ‘fsog‘
```

USB-Medium einbinden

Sei es aus Datensicherungsgründen oder weil man den RaspberryPi als Medien- bzw. Dateiserver betreiben will, benötigt man die Einbindung eines externen USB-Mediums. Ideal für den Betrieb am Pi sind USB-Sticks, da diese keine eigene Stromversorgung benötigen. Nachteil ist, daß diese Teile (zumindest Momentan) nur bis zu einer Größe von 64 GB zu einem halbwegs erschwinglichen Preis erhältlich sind.

USB-Festplatten können jedoch genauso verwendet werden, hier ist allerdings zu beachten, daß diese eben Strom brauchen. Entweder über USB (bei 2,5“) oder ein eigenes Netzteil (bei 3,5“).

Das Anbinden erfolgt in wenigen simplen Schritten:

Zunächst werden die benötigten Treiber installiert, damit NTFS und HFS+ Speichermedien eingebunden werden können.

```
sudo apt-get install ntfs-3g hfsutils hfsprogs
```

Danach wird ein Ordner im Verzeichnis */media* angelegt, in den das USB-Speichermedium später eingebunden wird (z.B. „usbstick“). Der Name kann frei gewählt werden, sollte aber keine Sonder- oder Leerzeichen enthalten.

```
sudo mkdir /media/usbstick
```

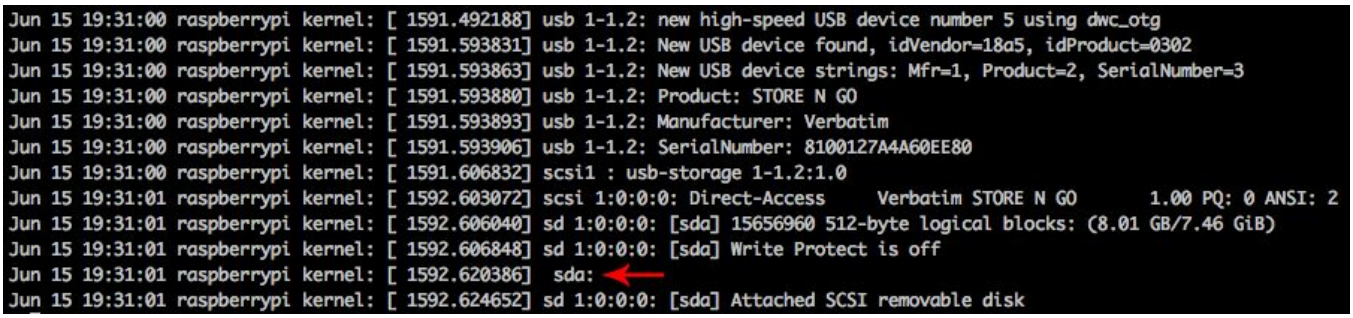
Nun wird der Befehl

```
tail -f /var/log/messages
```

in der Konsole ausgeführt, und anschließend das USB-Medium eingesteckt.

Daraufhin sollte in der Konsole eine ähnliche Ausgabe wie im Screenshot erscheinen.

Der rote Pfeil zeigt auf den Systemnamen des USB-Mediums, der für das Einhängen notwendig ist. Im Beispiel „sda“.



Nachdem der Name bekannt ist, kann die Ausgabe mittels STRG + C wieder verlassen/beendet werden.

Als letzter Schritt kann nun das externe Speichermedium eingehängt werden.

Dabei muss man je nach Dateisystem ein anderes Kommando verwenden, „sda“ durch den Namen des USB-Speichermediums und „/media/usbstick/“ durch den eigenen Mountpoint ersetzten.

Die zwei „pi“ Angaben müssen, sofern ein anderer Benutzer als *pi* auf das USB-Speichermedium zugreifen soll, durch dessen Namen ersetzt werden.

FAT32

```
sudo mount -t vfat -o uid=pi,gid=pi /dev/sda /media/usbstick/
```

NTFS

```
sudo mount -t ntfs-3g -o uid=pi,gid=pi /dev/sda /media/usbstick/
```

HFS+

```
sudo mount -t hfsplus -o force,rw,uid=pi,gid=pi /dev/sda /media/usbstick/
```

Zum Auswerfen des Datenträgers wird folgender Befehl verwendet:

```
sudo umount /media/usbstick
```

Soll der USB-Stick oder die USB-Festplatte nun bei jedem Start des RaspberryPis automatisch eingebunden werden, muss zunächst die sogenannte UUID ausgelesen werden.

Dies kann über folgendes Kommando bewerkstelligt werden.

```
sudo blkid /dev/sda
```

Nun muss man je nach Dateisystem einen der folgenden Befehle am Ende der Datei /etc/fstab eintragen.

Dabei muss sowohl die UUID als auch der Mountpoint angepasst werden. Dazu kann man den Editor „nano“ nutzen.

```
sudo nano -Bw /etc/fstab
```

FAT32

```
UUID=3241-40CE /media/usbstick/ vfat defaults,auto,umask=000,users,rw 0
```

NTFS

```
UUID=3241-40CE /media/usbstick/ ntfs-3g defaults,auto,umask=000,users,rw 0
```

HFS+

```
UUID=3241-40CE /media/usbstick/ hfsplus defaults,auto,umask=000,users,rw 0
```

Fertig! Es ist nun ein USB-Medium mit dem Dateisystem FAT32, NTFS oder HFS+ im RaspberryPi eingebunden und wird automatisch beim Start eingehängt.

Vergessenes Passwort zurücksetzen

Stecken Sie am PC die SD-Karte des Raspberry Pi in den Kartenleser ein, und öffnen Sie die Partition „boot“. Dort finden Sie im Wurzelverzeichnis die Datei „cmdline.txt,“ die Sie mit einem Texteditor öffnen.

Am Ende der Datei tragen Sie hinter einem Leerzeichen das zusätzliche Statement

```
init=/bin/sh
```

ein. Wichtig: Hängen Sie den Befehl am Ende an, schreiben Sie ihn aber nicht in eine neue Zeile. Nach dem Speichern der Datei booten Sie den Raspberry Pi wieder von der Karte und erhalten eine einfache Shell, die im root-Kontext läuft. Hier geben Sie den Befehl

```
passwd pi
```

ein, um das Passwort des Users „pi“ neu zu setzen. Mit den beiden Kommandos

```
sync
exec /sbin/init
```

fahren Sie mit dem Systemstart normal fort. Um die Änderung an „cmdline.txt“ wieder rückgängigzumachen, öffnen Sie mit root-Rechten beziehungsweise sudo die Datei unter „/boot/cmdline.txt“ im Texteditor nano und entfernen „init=/bin/sh“. Diese Änderung können Sie natürlich auch wieder an einem anderen PC mit Kartenleser durchführen. Ab jetzt lässt sich das Raspbian-System wieder normal starten, und die Anmeldung gelingt mit dem neuen Passwort. Der Trick klappt bei allen Raspberry-Pi-Distributionen, die auf Raspbian basieren, also auch mit Rasp BMC.

Backup der ownCloud bzw. Raspbian

Es ist immer eine gute Idee, seine Daten zu sichern.

Im Falle der ownCloud (bzw. Raspbian) kann man sich sogar selektives Sichern sparen und stattdessen (aufgrund der meist relativ geringen Größe der SD-Karte) einfach ein Image des gesamten Systems erstellen.

Image unter Windows erstellen

Zunächst muss das Tool Win32 Disk Imager installiert werden. Zu finden unter *http://sourceforge.net/projects/win32diskimager/files/latest/download*.

Im Win32 Disk Imager wird nun im Feld „Image File“ der Pfad samt Dateiname angegeben, wo das Image gespeichert werden soll.

Danach das Laufwerk, auf dem die SD-Karte eingebunden ist wählen und auf „Read“ klicken.

Image unter Linux bzw. Mac OS X erstellen

Unter Linux und Mac OS X ist es recht einfach mittels dd ein Abbild der SD-Karte zu erstellen.

Extrem wichtig ist beim Arbeiten mit dd, dass man immer genau wissen sollte, welche Gerätedateien man verwendet!

Also lieber einmal mehr Kontrollieren (z.B. mit blkid), damit man nicht versehentlich die interne Festplatte überbügelt...

Dabei muss man */dev/sdx* durch den tatsächlichen Gerätenamen der SD-Karte ersetzen (bei Mac OS X für gewöhnlich /dev/diskx), sowie */pfad/dateiname.img* durch den Pfad samt Dateinamen wo das Abbild gespeichert werden soll.

Für den Dateinamen bietet sich etwas halbwegs „sprechendes“ an, damit man bei mehreren Abbildern nicht den Überblick verliert.

Zum Beispiel: „raspbian_owncloud_150414.img“.

```
dd if=/dev/sdx of=/pfad/dateiname.img bs=1M
```

Die Imagedatei kann man aus Platzgründen auch schon beim Anlegen komprimieren:

```
dd if=/dev/sdx | gzip > /pfad/dateiname.img.gz
```

Fertig! Nun ist der aktuelle Stand des Betriebssystems und der installierten ownCloud gesichert, und man kann unbesorgt neue Experimente ausprobieren und im Bedarfsfall einfach das gesicherte Abbild wieder einspielen. Das erfolgt mit den Befehlen:

```
dd if=/pfad/dateiname.img of=/dev/sdx
```

bzw.

```
gunzip -c /pfad/dateiname.img.gz | sudo dd of=/dev/sdx
```

Falls benötigt, kann man das Abbild auch in einem anderen Linux-System als Gerät einbinden.

Dazu benötigt man einen Einhängepunkt (z.B. /media/loop_mount) und kann danach das erzeugte Abbild direkt einhängen:

```
sudo mount -o loop /pfad/dateiname.img /media/loop_mount
```