OwnCloud 6 auf Debian

Vorüberlegungen

Ich hatte ja bereits darüber berichtet, dass ich aus verschiedenen Gründen meine ownCloud-Instanz nicht mehr auf der Synology Diskstation halte. Die Querelen die mit der neuen DSM 5 entstanden sind, zeigen die Richtigkeit meiner Entscheidung … Der seit Oktober 2013 als ownCloud-Server eingesetzte Raspberry Pi verichtet zwar klaglos seinen Dienst, ist aber auf Grund seiner Hardwareressourcen (insbesondere der Einsatz einer SD-Card als Speichermedium) inzwischen hart an der Grenze seiner Leistungsfähigkeit angekommen. Zudem hat die Himbeere ab und an Probleme, die mit dem installierten Workaround (Stichwort Watchdog) zwar minimiert werden konnten, partiell aber immer wieder auftreten. Zeit für Veränderungen also …

Schritt 1 - Auswahl der Hardware

Am Ende entschied ich mich für einen ***HP Proliant G7 Microserver N54L*** mit AMD Turion II N54L Prozessor (DualCore 2,2 GHz), 2 GB RAM, 1x250GB HDD, 1 GBit-Netzwerkkarte und einem 150W-Netzteil. Der Rechner bekommt gute Kritiken von den Kunden und der Preis liegt um die 170 – 180 €, also deutlich geringer als bei HP angegeben. Die Komponenten lassen eine erhebliche Performancesteigerung erwarten, zumal das System durchaus erweiterbar ist und im Fall des Falles problemlos RAM und Festplatten nachrüstbar sind. Für den Anfang sollte die angegebene Konfiguration jedoch mehr als ausreichend sein. Eine wichtige Funktion wird das konfigurierbare Wake on LAN der Maschine sein, da ich ohnehin keine permanente Verbindung zu ownCloud brauche, sonder per VPN nur bei Bedarf auf meine Wolke zugreife …

Schritt 2 Auswahl des Betriebssystems

Hier fiel meine Wahl auf Debian. Ein Linux war klar, meine Erfahrungen konzentrieren sich zudem ohnehin auf debianbasierte Distributionen. Hinzu kommt, dass die Quellen von ownCloud als Repository problemlos (auch) in Debian integriert werden können, somit jederzeit der von ownCloud heraus gegebene aktuelle Softwarestand verfügbar ist.

Das Gerät ist gestern bei mir eingetroffen, allerdings werde ich erst morgen Abend dazu kommen, die Installation vorzunehmen. An sich sollte die ganze Sache in etwa 3 Stunden zu schaffen sein. Warten wir es ab, ob es tatsächlich beim veranschlagten Zeitrahmen bleibt.

www.kusawe.de

Realisierung

Ja so schnell kann es gehen. Völlig überraschend für mich “öffnete sich gestern Abend ein Zeitfenster” (Zitat einer nicht näher genannt werden wollenden Person) und ich hatte Gelegenheit mich an die Verwirklichung des Projektes zu machen … Zunächst musste ich allerdings feststellen, dass ich meinen Rechner zu Hause “verkonfiguriert” hatte und so sah es in meinem Wohnzimmer am Ende aus wie an meinem Arbeitsplatz – 3 Rechner liefen (mehr oder weniger) und auf den diversen Bildschirme waren Konsolen zu sehen, die durch die dezente Farbgebung auffielen: nichts als Schwarz (der Hintergrund), rot, grün und weiß (die Schrift). Kurz gesagt, ich war in meinem Element .

Nach etwa 2 Stunden war der PC wieder im Vollbesitz seiner geistigen Kräfte und ich konnte mich meinem eigentlichen Vorhaben widmen …

Schritt1: Installation von Debian

Der HP Proliant G7 fiel bereits beim Auspacken durch seine kompakte Bauweise auf. Der Rechner war schnell aufgebaut und angeschlossen – mehr als Netzwerk und Stromanschluss sind für den täglichen Betrieb ja nicht nötig. Für die Installationsprozedur waren zusätzlich Tastatur, Monitor und ein USB-CD-Laufwerk erforderlich. Das benötigte Installationsmedium (Debian 7.4, Installation über das Netz) war schnell von der Seite des Debianprojektes geladen und gebrannt (dafür brauchte ich meinen PC). Der Proliant bootete klaglos von der CD und die Installation ging zügig vonstatten, sämtliche Hardwarekomponenten wurden problemlos erkannt. Da ich den Server ausschließlich für ownCloud verwenden will, genügte mir die Einrichtung einer Partition die die gesamte Plattenkapazität umfasst.
An Softwarekomponenten wurde neben dem Basissystem lediglich die Option “Webserver” ausgewählt, denn genau dass soll der Proliant am Ende ja sein. Nach der recht zügigen Installation der ausgewählten Optionen und dem obligatorischen Upgrade folgte

Schritt 2: Installation von Webmin.

Meine ausgesprochen positiven Erfahrungen mit diesem Tool, insbesondere was die Einrichtung der für mich zwingend notwendigen Datensicherung betrifft, führten zu dieser Installation. Hier sind dann doch einige Installationsschritte erforderlich (natürlich als root ausführen):

- Installation einiger Programme die Webmin benötigt

```
apt-get install perl libnet-ssleay-perl openssl libauthen-pam-perl libpam-
```

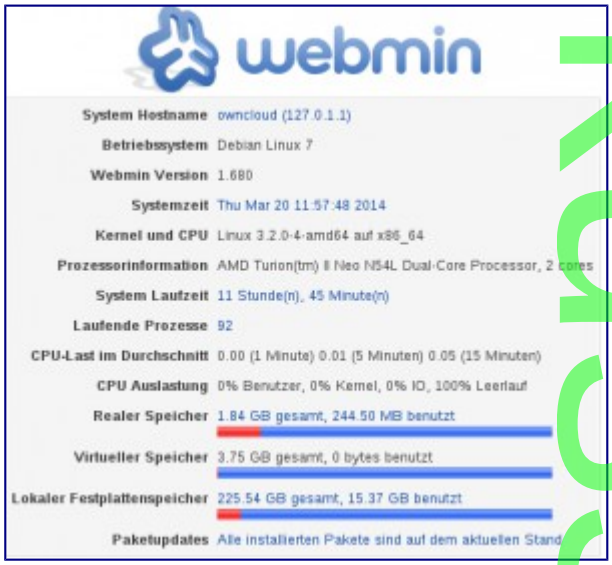
www.kuschelpolo.de

```
runtime libio-pty-perl apt-show-versions python
```

- Download und Installation von Webmin. Version 1.680 ist aktuell. Man kann aber auch eine vorherige Version installieren. Webmin sucht nach dem Start nach Updates und installiert diese gegebenenfalls

```
wget -c http://prdownloads.sourceforge.net/webadmin/webmin_1.680_all.deb
dpkg -i webmin_1.680_all.deb
```

Fertig. Nach der Anmeldung (https://IP_Adresse:10000) meldet sich Webmin wie gewohnt mit den Systeminformationen. Auf dem Bild ist diese Meldung bereits nach erfolgter Umstellung der Sprache auf deutsch zu sehen …



Schritt 3: Installation von ownCloud auf dem Debianserver

Als Anleitung kann euch hier das Administrationshandbuch von ownCloud.org dienen. Die entsprechenden Handbücher findet ihr auf der ***Webseite von owncloud***. Da Debian eine vom OpenSuse Build Service unterstützte Distribution ist, können wir die Quellen direkt in unsere Repositieris einpflegen.

```
echo 'deb
http://download.opensuse.org/repositories/isv:/ownCloud:/community/Debian_7.
0/ /' >> /etc/apt/sources.list.d/owncloud.list
apt-get update
apt-get install owncloud
```

Es ist empfehlenswert, auch den Schlüssel der Quelle ebenfalls zu installieren (spätere Updates werden durch das System als sicher eingestuft).

```
wget
```

www.kuss.de

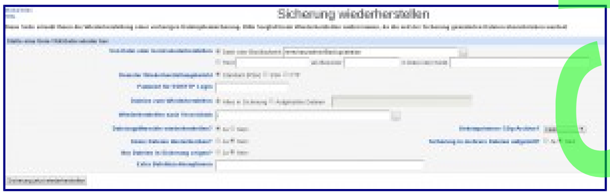
```
http://download.opensuse.org/repositories/isv:ownCloud:community/Debian_7.0/
Release.key
apt-key add - < Release.key
```

Nach erfolgter Installation, können wir nun erstmals unsere Wolke starten um die grundlegende Verzeichnisstruktur anlegen zu lassen. Ich verwende als Datenbank SQLite, da ich allein auf owncloud zugreife. In Mehrbenutzerumgebungen empfiehlt ownCloud.org die Verwendung von MySQL.

Schritt 4: Rücksicherung der Daten

Die Vorgehensweise der Einrichtung eines Backups aus Webmin heraus habe ich ***bereits hier im Blog beschrieben*** (Link zum Download). Die ersten Schritte – Konfiguration des Zugriffs auf die Diskstation usw. – sind wie dort beschrieben vorzunehmen. Da der Ablauf gleich ist, erspare ich mir hier weitere Ausführungen …

Die Rücksicherung ist ebenfalls denkbar einfach: im Bild seht ihr die Einstellungen, die ich beim Rücksicherungsauftrag vorgenommen habe. Damit werden die entsprechenden Dateien zurück in das Standardverzeichnis der ownCloud-Installation (/var/web/owncloud) gesichert.



Schritt 5: Entpacken des Dump der Datenbank und Anpassen der Systemrechte

Unsere Daten befinden sich jetzt an Ort und Stelle, allerdings müssen noch ein paar Kleinigkeiten erledigt werden. Zunächst muss sqlite3 installiert werden, um aus unserem Dump wieder eine lauffähige Datenbank zu machen:

```
apt-get install sqlite3
```

Weiterhin muss nunmehr die Datenbank aus dem Dump hergestellt werden (vorher die existierende Datenbankdatei löschen) und Dateisystemrechte angepasst werden:

```
sqlite3 /var/www/owncloud/data/owncloud.db <
/var/www/owncloud/data/admin/ownclouddb.bak
chown -R www-data:www-data /var/www/owncloud/admin
chown -R www-data:www-data /var/www/owncloud/admin/data
```

Jetzt noch ein Modul zum Rewrite von Urls im Apache aktivieren und den Webserver neu starten:

```
a2enmod rewrite
```

```
service apache2 restart
```

Fertig! Der Zugriff auf meine ownCloud-Instanz mit sämtlichen Daten inklusive Kontakt- und Kalenderdaten funktionierte reibungslos. Da beim Restore auch das Verzeichnis Config zurück gesichert wurde, sind auch sämtliche Konfigurationseinstellungen übernommen worden.

Fazit:

Ein wenig Aufwand war nötig, aber mein Ziel habe ich erreicht: Der Zugriff auf owncloud – auch und gerade auf das Webinterface sowohl lokal als auch per VPN – wurde deutlich beschleunigt. Natürlich waren noch einige Anpassungsschritte erforderlich u.a. Einrichtung eines Backups, Definition des Cron-Jobs (siehe Beitrag Pimp my owncloud), aber die immanente Performancesteigerung war die Mühen wert.

Pimp my ownCloud

Auch wenn ownCloud nunmehr in einer sehr zufriedenstellenden Geschwindigkeit auf dem HP Proliant MicroServer läuft, bleiben einige kleine Schritte zu tun, um die Performance weiter zu optimieren.

Im Netz findet ihr diverse Tuningtipps. Auf 3 davon habe ich mehrfach in Artikeln hier auf dem Blog hingewiesen. Da sich aber nunmehr die Umgebung meiner Installation (wieder einmal) grundlegend geändert hat, möchte ich die Umsetzung dieser Schritte für eben diese Umgebung darstellen – nicht zuletzt um später selbst noch einmal nachlesen zu können).

Schritt 1 – Cronjobs

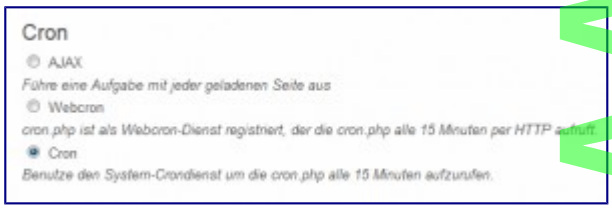
Im Administrationshandbuch von ownCloud 6 findet man zu Cronjobs folgendes:

“Owncloud benötigt verschiedene automatisch ausgeführte Hintergrundjobs. Es gibt 3 Methoden sie zu realisieren. Standardweg ist AJAX und der empfohlene Weg ist cron.”

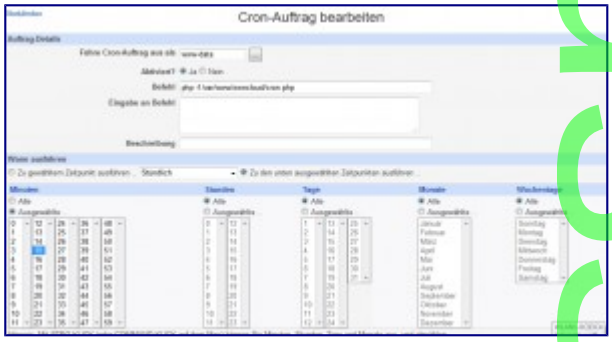
Die Verwendung des Cron-Daemons des zugrundeliegenden Systems wird empfohlen, da auf diesem Wege eventuell vorhandene Restriktionen des verwendeten Web-Servers nicht wirksam werden.

Die erforderlichen Schritte sind recht einfach umzusetzen. Zuerst wird im Administrationsbereich von ownCloud die Einstellung von AJAX auf Cron geändert. Das ist mit einem Mausklick getan.

www.kuschel.de



Nunmehr muss im System ein Cron-Job definiert werden, der die im ownCloud-Verzeichnis liegende cron.php in regelmäßigen Abständen aufruft. Im Administrationshandbuch wird empfohlen, die cron.php alle 15 Minuten aufzurufen. Wie immer gibt es natürlich die Möglichkeit, den entsprechenden Job per Konsole zu erstellen, ich nutze aber gern und häufig Webmin für solche Aufgaben. In Webmin finden wir im Menu System den Punkt “Geplante Aufträge (Cron)”. Auf der damit aufgerufenen Seite werden bereits bestehende Cron-Jobs angezeigt die ich bearbeiten, aktivieren, deaktivieren oder löschen kann. Hier kann ich aber auch neue Aufträge definieren. Im Bild seht ihr die Einstellungen für unseren geplanten Cron-Job: der Nutzer www-data (Standardbenutzer des Apache-Webservers) ruft alle 15 Minuten die Datei cron.php im Owncloudverzeichnis über den Befehl **php –f /var/www/owncloud/cron.php** auf. Speichern und aktivieren – fertig!



Schritt 2 – Caching aktivieren

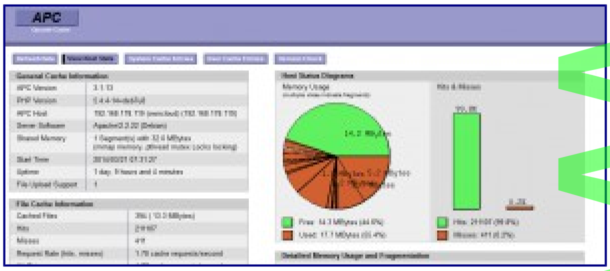
Eine deutliche Beschleunigung des Webinterfaces von ownCloud in der Vergangenheit brachte bei mir die Installation des php-Cachingmoduls. Zur Installation bevorzuge ich dieses Mal die Konsole: als root

apt-get install php-apc

und das Modul wird installiert. Zusammen mit dem Modul wird ein Skript installiert, dass über den Zusatnd des Moduls und dem Cacheverhalten aufklärt. Im Verzeichnis **/usr/share/doc/php-apc** liegt die Datei **apc.php**. Durch das Setzen eines Links auf die Datei im Verzeichnis unseres ownCloud-Servers, können wir die entsprechenden Informationen direkt über unseren Webserver abrufen. Mithilfe des Datei-Managers in Webmin (im Menu unter “Sonstiges” zu finden) ist der Link schnell definiert und per http://IP-Adresse/owncloud/apc.php aufrufbar. Einen Auszug der

www.apfelkiste.de

Informationen seht ihr im Bild.



Ebenfalls sehr hilfreich, wenn auch nicht zum pimpen von ownCloud erforderlich, ist die Ansicht der php-Informationen. Dazu wird im Vereichnis /var/www/owncloud eine Datei phpinfo.php erzeugt, die lediglich eine Zeile enthält: <? phpinfo(); ?>. Per http://IP_Adresse/owncloud/phpinfo.php aufgerufen werden wir über php-Version unseres Systems, geladene Module und einiges anderes informiert.

PHP Version 5.4.4-14+deb7u8

System	Linux owncloud 3.2.0-4-amd64 #1 SMP Debian 3.2.54-2 x86_64
Build Date	Feb 17 2014 09:19:04
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-mysqlnd.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-apc.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-intl.ini, /etc/php5/apache2/conf.d/20-mcrypt.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-pdo_pgsql.ini, /etc/php5/apache2/conf.d/20-pdo_sqlite.ini, /etc/php5/apache2/conf.d/20-pgsql.ini, /etc/php5/apache2/conf.d/20-sqlite3.ini, /etc/php5/apache2/conf.d/imagick.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS

Schritt 3 – nicht benötigte Apps deaktivieren

Selbsterklärend, trotzdem nicht unwichtig. Wer keine Musik oder Bilder auf seiner ownCloud-Instanz speichert, benötigt auch die entsprechenden Appsnicht. Da sich die Anwendungen einfach per Mausklick aktivieren oder deaktivieren lassen, verbaut man sich auch zukünftig nicht die Nutzung der entsprechenden Anwendungen. Also alles was nicht unmittelbar benötigt bzw. genutzt wird – deaktivieren.

Für die Zukunft

Im Moment läuft ownCloud bei mir auf Basis von SQLite. Eine Migration auf MySQL (bzw. MariaDB) scheint derzeit nichtmöglich zu sein. Da ich allein auf meinen ownCloud-Server zugreife, bisher auch kein Problem. In Mehrbenutzerumgebungen empfiehlt ownCloud.org jedoch MySQL. Bei Bedarf hätte ich gern die Möglichkeit der Migration. Die Programmierer scheinen

fieberhaft an einem Migrationsskript zu arbeiten, ohne bisher zu einem zufriedenstellenden Ergebnis gekommen zu sein …

Üben wir uns ein wenig in Geduld.

www.kussam.de