Cloudlösung Seafile CE auf dem Raspberry Pi einrichten

Workshop zur Einrichtung und Installation von Seafile CE im privaten Netzwerk, mit externer Anbindung über HTTPS.

Marek Walther

2016-04-09

Hardware Freedom Day 2016 Kiel

Inhaltsverzeichnis

Ziel des Workshops

Nach Durchführung dieses Workshops sollte der Teilnehmer in der Lage sein, einen Raspberry Pi neu aufzusetzen und die Cloudsoftware Seafile zu installieren und einzurichten. Hierzu gehört auch sichere Konfiguration von SSH mit Zertifikaten und den Cloudzugriff über einen SSH-Tunnel.

Benötigtes Workshopmaterial

Für die Durchführung dieses Workshops werden folgende Materialien benötigt:

- Raspberry Pi in der Ausführung B mit 512MB RAM und Netzteil
    - Besser ein Raspberry Pi 2 Model B oder Raspberry Pi 3 Model B mit 1GB RAM und Netzteil
- SD-Speicherkarte mit einer Kapazität von mindestens 4GB
- USB-Stick mit einer Kapazität von mindestens 4GB als Datenspeicher
- Netzwerkkabel
- Monitor und Tastatur für eine eventuelle Fehlersuche
- Rechner mit Linux (Debian / Ubuntu)
- Zugriff auf das Internet

Grundinstallation und Einrichtung des Raspberry Pi

Die Grundinstallation des Raspberry Pi stellt eine stabile Rechnerumgebung zur Verfügung, auf der Serverdienste installiert und betrieben werden können. Generell ist es sinnvoll bei einer Serverinstallation immer mit einer sauberen Installation oder einer Installationsvorlage zu beginnen.

Dieser Abschnitt umfasst folgende Schritte:

1. Anlegen eines Arbeitsbereiches auf dem Konfigurationscomputer.
2. Vorbereiten der SD-Speicherkarte und Installation einer passenden Raspberry Pi Distribution.
3. Initialisierung der Installation.
4. Einrichten einer statischen Netzwerkkonfiguration mit Internetzugriff.
5. Updaten des Betriebssystems über das Internet.
6. Vorbereiten und Einbinden eines USB-Sticks als Datenspeicher.

Anlegen eines Arbeitsbereichs auf dem Konfigurationscomputer

Zum sauberen Arbeiten und als Ergebnissicherung sollte zum Beginn des Workshops ein eigenes Arbeitsverzeichnis auf dem verwendeten Konfigurationscomputer angelegt werden. Aus diesem Verzeichnis sollten alle Arbeiten durchgeführt und alle getätigten Downloads und erstellten Konfigurationsdateien abgelegt werden.

Für diesen Schritt rufen Sie bitte eine Eingabekonsole nach Art ihrer verwendeten Distribution auf. Danach erstellen Sie im Heimatverzeichnis einen neuen Ordner und wechseln in den neuen Ordner. Zum Abschluss wir das aktuelle Arbeitsverzeichnis geprüft.

```
mawa@debian:~$ mkdir ~/workshop
mawa@debian:~$ cd ~/workshop
mawa@debian:~/workshop$ pwd
/home/mawa/workshop
```

Befinden Sie sich nach dem Verzeichniswechsel im richtigen Verzeichnis, kann es weiter gehen.

Vorbereiten der SD-Speicherkarte

Im zweiten Schritt muss die SD-Speicherkarte vorbereitet und mit einem passenden Betriebssystem versehen werden. Hierbei ist zu klären, welches Betriebssystem zum Einsatz kommen soll. Zur Auswahl steht neben dem Standardbetriebssystem *Raspbian* auch ein auf den Serverbetrieb optimiertes Betriebssystem mit dem Namen *Minibian*.

Raspbian ist auf dem Raspberry Pi das wohl am meisten verwendete Betriebssystem, da es vom Raspberry Projekt als Standard auserkoren wurde. Raspbian hat den Vorteil, dass es eine grafische Benutzeroberfläche mitliefert, an der man sich zur Not festhalten kann. Dafür frisst diese Oberfläche wertvolle Ressourcen wie SD-Kartenspeicher und Hauptspeicher. Die Mindestgröße für die SD-Karte beträgt hier 4GB.

Seit Neuem gibt es von Raspian auch eine Lite Version, die mit abgespeckter vorinstallierter Software daher kommt. Für einen reinen Servierdienst ohne weitreichende Desktopanwendungen und als Basis für Eigenentwicklungen ist diese besser geeignet.

Installation von Raspbian

Das Raspbian-Image für eine 4GB SD-Karte erhält man über die Downloadseite[1] des Raspberry Pi Projektes. Den Download erledigt man am besten mit dem Werkzeug *wget* und nutzt dabei den Link für den Zip-Download.

```
mawa@debian:~/workshop$ wget https://downloads.raspberrypi.org/raspbian_lite_latest
--2016-04-08 20:47:29--  https://downloads.raspberrypi.org/raspbian_lite_latest
Auflösen des Hostnamen »downloads.raspberrypi.org (downloads.raspberrypi.org)«...
93.93.130.214, 93.93.130.39, 93.93.128.230, ...
Verbindungsaufbau zu downloads.raspberrypi.org (downloads.raspberrypi.org)|
93.93.130.214|:443... verbunden.
HTTP-Anforderung gesendet, warte auf Antworlt... 302 Found
Platz: https://downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2016-
03-18/2016-03-18-raspbian-jessie-lite.zip[folge]
--2016-04-08 20:47:30--
https://downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2016-03-
18/2016-03-18-raspbian-jessie-lite.zip
Wiederverwendung der bestehenden Verbindung zu downloads.raspberrypi.org:443.
HTTP-Anforderung gesendet, warte auf Antwort... 302 Found
Platz:
http://director.downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2016-
03-18/2016-03-18-raspbian-jessie-lite.zip[folge]
--2016-04-08 20:47:30--
http://director.downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2016-
03-18/2016-03-18-raspbian-jessie-lite.zip
Auflösen des Hostnamen »director.downloads.raspberrypi.org
(director.downloads.raspberrypi.org)«... 93.93.130.214, 93.93.130.39,
93.93.128.230, ...
Verbindungsaufbau zu director.downloads.raspberrypi.org
(director.downloads.raspberrypi.org)|93.93.130.214|:80... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 298225874 (284M) [application/zip]
In »»raspbian_lite_latest«« speichern.

raspbian_lite_latest   100%[=============================>] 284,41M  3,29MB/s   in
56s

2016-04-08 20:48:25 (5,12 MB/s) - »»raspbian_lite_latest«« gespeichert
[298225874/298225874]

mawa@debian:~/workshop$
```

Nach dem vollständigen Herunterladen muss das komprimierte Image ausgepackt werden. Hierfür kommt das Werkzeug *unzip* zum Einsatz. Vorher muss die Datei noch unbenannt werden, denn unzip entpackt nur Dateien mit der Endung .zip.

```
mawa@debian:~/workshop$ mv raspbian_lite_latest raspbian_lite_latest.zip
mawa@debian:~/workshop$ unzip raspbian_lite_latest.zip
Archive:  raspbian_lite_latest.zip
  inflating: 2016-03-18-raspbian-jessie-lite.img
mawa@debian:~/workshop$ ls -lh
insgesamt 1,6G
-rw-r--r-- 1 mawa mawa 1,3G Mär 18 08:17 2016-03-18-raspbian-jessie-lite.img
-rw-r--r-- 1 mawa mawa 285M Mär 18 09:17 raspbian_lite_latest.zip
mawa@debian:~/workshop$
```

Nach dem Auspacken liegt das Image unter einem neuen Dateinamen im Arbeitsverzeichnis. Eine Auflistung des Arbeitsverzeichnisses zeigt, das jetzt 1,6 GB an Dateien auf unserer Festplatte angesammelt haben.

1 http://www.raspberrypi.org/downloads/

Jetzt ist es soweit, die Speicherkarte über einen Kartenleser an den Konfigurationsrechner anzubinden. Hierbei ist es wichtig, dass das verwendete Betriebssystem die Speicherkarte nicht automatisch einbindet. Ggf. muss die Speicherkarte nach dem Einsetzen manuell nach Art des Betriebssystems wieder ausgebunden werden.

Nach dem Einsetzen der SD-Speicherkarte gilt es herauszufinden, als welches Gerät die Speicherkarte von unserem Betriebssystem erkannt wurde. Dieses können wir mit dem Werkzeug *dmesg* herausfinden.

```
mawa@debian:~/workshop$ dmesg
[ 4813.131000] usb 3-1.2: new high-speed USB device number 5 using ehci_hcd
[ 4813.237397] usb 3-1.2: New USB device found, idVendor=05e3, idProduct=0723
[ 4813.237407] usb 3-1.2: New USB device strings: Mfr=3, Product=4, SerialNumber=2
[ 4813.237413] usb 3-1.2: Product: USB Storage
[ 4813.237418] usb 3-1.2: Manufacturer: Generic
[ 4813.237421] usb 3-1.2: SerialNumber: 000000009451
[ 4813.772170] Initializing USB Mass Storage driver...
[ 4813.772486] usb-storage 3-1.2:1.0: Quirks match for vid 05e3 pid 0723: 8000
[ 4813.772547] scsi6 : usb-storage 3-1.2:1.0
[ 4813.772771] usbcore: registered new interface driver usb-storage
[ 4813.772776] USB Mass Storage support registered.
[ 4814.770897] scsi 6:0:0:0: Direct-Access     Generic  STORAGE DEVICE   9451 PQ:
0 ANSI: 0
[ 4814.773116] sd 6:0:0:0: Attached scsi generic sg1 type 0
[ 4814.778448] sd 6:0:0:0: [sdb] Attached SCSI removable disk
[ 4832.916364] sd 6:0:0:0: [sdb] 15564800 512-byte logical blocks: (7.96 GB/7.42
GiB)
[ 4832.920309] sd 6:0:0:0: [sdb] No Caching mode page found
[ 4832.920319] sd 6:0:0:0: [sdb] Assuming drive cache: write through
[ 4832.928227] sd 6:0:0:0: [sdb] No Caching mode page found
[ 4832.928237] sd 6:0:0:0: [sdb] Assuming drive cache: write through
[ 4832.929885]  sdb: sdb1
mawa@debian:~/workshop$
```

Die Logausgabe des Werkzeugs gibt Auskunft darüber, dass die Beispielkarte als Gerät sdb erkannt wurde. Bitte stellen Sie sicher, das Sie das richtige Gerät verwenden und gleichen Sie zusätzlich die Größe erkannten Speicherplatzes mit dem Speicherplatz der verwendeten Speicherkarte ab.

Zur Sicherheit sollten Sie jetzt zusätzlich die auf der Speicherkarte erkannte Partition demontieren, damit diese nicht vom Betriebssystem eingebunden und verwendet wird. Hierbei ist zu beachten, dass die Geräte über das Geräteverzeichnis /dev angesprochen werden.

```
mawa@debian:~/workshop$ umount /dev/sdb1
umount: /dev/sdb1 is not mounted (according to mtab)
mawa@debian:~/workshop$
```

Die beim Demontieren auftretende Fehlermeldung gibt Auskunft darüber, dass das Gerät noch nicht eingebunden war. Das ist in unserem Falle OK und stellt kein Problem dar. Kommt eine Fehlermeldung, dass das Gerät nicht ausgebunden werden kann, weil es derzeitig in Verwendung ist, ist zu prüfen, welche Prozesse dieses Gerät blockieren.

Jetzt ist es soweit, das Image kann geschrieben werden, wobei es jetzt Zeit für eine Warnung ist.

WARNUNG

Beim Schreiben des Images gehen alle Daten auf dem beschrieben Gerät verloren. Daher prüfen Sie bitte vorher noch einmal genau, ob wirklich das richtige Gerät und die zugehöhrige Gerätedatei verwendet werden. Das Beschreiben der falschen Gerätedatei kann zu Datenverlust und Startproblemen auf dem Konfigurationsrecher führen.

Schreiben Sie daher *nicht* einfach die Parameter dieser Beispiel-installation ab, sondern ermitteln Sie die für Sie gültigen Parameter!

Wenn alles OK ist, kann mit dem Werkzeug *dd* die ausgepackte Imagedatei auf die Speicherkarte geschrieben werden. Da für den Blockgerätezugriff administrative Rechte notwendig sind, wird das Werkzeug mittels des Werkzeugs *sudo* (*set user and do*) ausgeführt. Hierbei kann es notwendig sein, das eigene Benutzerkennwort einzugeben. Das Schreiben des Images mit seinen 3GB kann eine längere Zeit in Anspruch nehmen.

```
mawa@debian:~/workshop$ sudo dd bs=4M if=2016-03-18-raspbian-jessie-lite.img
of=/dev/sdb
[sudo] password for mawa:
324+1 Datensätze ein
324+1 Datensätze aus
1361051648 Bytes (1,4 GB) kopiert, 165,261 s, 8,2 MB/s
mawa@debian:~/workshop$
```

Zum Abschluss dieses Arbeitsschrittes wird das Betriebssystem noch angewiesen, im Speicher gecachte Datenblöcke auf das Gerät zu schreiben.

```
mawa@debian:~/workshop$ sudo sync
[sudo] password for mawa:
mawa@debian:~/workshop$
```

Danach kann die Speicherkarte aus dem Kartenleser entfernt und in den Raspberry eingesetzt oder das Image auf dem Arbeitsrechner eingebunden und bearbeitet werden.

Initialisierung des Betriebssystems

Nach der Installation des Images auf der SD-Speicherkarte muss das Betriebssystem initialisiert werden. Der normale Weg besteht darin, den Raspberry mit der neu beschriebenen SD-Card zu booten und die Konfiguration über das Initialisierungsmenü durchzuführen. Hierbei werden Monitor, Tastatur und viel Zeit benötigt.

Um dieses zu umgehen, besteht ein weiterer Weg darin die SD-Card auf einem Linuxsystem einzubinden, den Initialisierungsprozess zu deaktivieren, eine feste IP-Adresse zu konfigurieren und den SSH-Server zu initialisieren. Danach kann die SD-Karte in den Raspberry eingesteckt und gestartet werden. Der Zugriff auf den Rasperry ist danach über das Netzwerk mittels dem Werkzeug ssh möglich und der Initialisierungsprozess kann dann über die Befehlszeile manuell gestartet werden.

Vorbereitung der Initialisierung am Konfigurationscomputer

Im ersten Schritt sollte hier die SD-Card aus dem Konfigurationsrecher entfernt und neu eingesetzt werden, damit die neuen Partitionen sicher erkannt werden. Danach erfolgt die Vorbereitung mittels folgender Schritte:

1. Erstellen eines Mountpoints an dem das Dateisystem, der SD-Card, montiert werden kann.

2. Montieren des Dateisystems am Mountpoint.

3. Deaktivieren des Raspbian Autoinitialisierungsprozesses.

4. Erstellen der fehlenden SSH-Hostschlüssel.

5. Netzwerkkonfiguration mit einer festen IP-Adresse.

6. Demontieren des Dateisstems vom Mountpoint.

Ein Mountpoint ist schnell erstellt, es handelt sich hierbei um ein leeres Verzeichnis, das als Unterverzeichnis im Projektordner mit dem Werkzeug *mkdir* erstellt wird.

```
mawa@debian:~/workshop$ mkdir raspberry
```

Für den nächsten Schritt gilt es die Systempartition auf der SD-Card zu ermitteln. Hier kann es helfen, sich die vom Kernel erkannten Partitionen anzuschauen.

```
mawa@debian:~/workshop$ cat /proc/partitions
major minor  #blocks  name

   8        0  488386584 sda
   8        1     307200 sda1
   8        2     102400 sda2
   8        3     131072 sda3
   8        4  255459328 sda4
   8        5  102400000 sda5
   8        6  117760000 sda6
   8        7     976896 sda7
 254        0  117757952 dm-0
 254        1   29294592 dm-1
 254        2   48824320 dm-2
 254        3   14647296 dm-3
   8       16    7761920 sdb
   8       17      57344 sdb1
   8       18    3138560 sdb2
```

Da auch schon beim Einspielen des Raspbianimages die SD-Card auf dem Laufwerk sdb lag, liegen jetzt auch die Partitionen unter diesem Laufwerk. Die SD-Card enthält zwei Partitionen. Die kleinere Partition mit ca. 60 MB enthält eine FAT-Partition mit einem Bootloader und etwas Kleinkram. Das System liegt auf der zweiten Partition und hat eine Größe von etwas mehr als 1,2 GB und wird mittels dem Werkzeug *mount* am vorbereiteten Mountpoint montiert..

```
mawa@debian:~/workshop$ sudo mount /dev/sdb2 raspberry
```

Ein kurzes Auflisten des montierten Verzeichnisses sollte dann auch die Struktur eines Wurzellaufwerkes mit seiner FHS-Struktur offenbaren.

```
mawa@debian:~/workshop$ ls raspberry/
bin  dev  home  lost+found mnt proc run  selinux sys usr
boot etc  lib   media      opt root sbin srv     tmp var
```

Anmerkung: Die nachfolgenden Schritte zur Deaktivierung der Erstkonfiguration beziehen sich noch auf die alte Version unter Debian Wheezy, unter Jessie sind diese nicht notwendig und so auch nicht mehr durchzuführen.

Ist das der Fall, kann jetzt der Initialisierungsprozess von Raspbian deaktiviert werden. Hierfür ist zuerst ein Startscript zu löschen.

```
mawa@debian:~/workshop$ sudo rm raspberry/etc/profile.d/raspi-config.sh
```

Danach ist die Konfiguration des Init-Prozesses mit dem Texteditor *nano* anzupassen.

```
mawa@debian:~/workshop$ sudo nano raspberry/etc/inittab
```

In etwa der Mitte der Konfigurationsdatei befinden sich zwei Konfigurationszeilen, von denen eine auskommentiert ist.

```
[...]
#1:2345:respawn:/sbin/getty --noclear 38400 tty1 # RPICFG_TO_ENABLE
1:2345:respawn:/bin/login -f root tty1 </dev/tty1 >/dev/tty1 2>&1 # RPICFG_TO_DISABLE
[...]
```

Die Auskommentierung der Zeilen ist jetzt entsprechen umzukehren, damit nach dem Start nur ein normaler getty-Prozess an der TTY-Schnittstelle lauscht.

```
[...]
1:2345:respawn:/sbin/getty --noclear 38400 tty1 # RPICFG_TO_ENABLE
#1:2345:respawn:/bin/login -f root tty1 </dev/tty1 >/dev/tty1 2>&1 # RPICFG_TO_DISABLE
[...]
```

Nach der Änderung kann die Datei mit der Tastenkombination *<STRG>+O* gespeichert und der Texteditor mit der Tastenkombination *<STRG>+X* geschlossen werden. Damit ist die Erstinitialisierung zum ersten Neustart der Raspberry deaktiviert und Rechnern startet direkt mit einem Login.

Danach ist es notwendig dafür zu sorgen, dass der SSH-Server auf der Raspbian Installation aktiviert und gestartet wird. Um dieses zu erreichen, müssen jetzt manuell die SSH-Hostkeys erstellt werden. Hierfür wird das Werkzeug *ssh-keygen* verwendet und es ist darauf zu achten, das die Keys im richtigen Unterverzeichnis erstellt werden.

```
mawa@debian:~/workshop$ sudo ssh-keygen -t rsa -b 2048 -f raspberry/etc/ssh/ssh_host_rsa_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in raspberry/etc/ssh/ssh_host_rsa_key.
Your public key has been saved in raspberry/etc/ssh/ssh_host_rsa_key.pub.
The key fingerprint is:
99:4d:b6:8e:6d:fa:dc:54:3a:52:ec:0d:0d:fb:e6:71 root@debian
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|          o .    |
|         * o +   |
|        S o = o  |
|         + o *   |
|        . = = = E|
|         + + + o |
|        ..o . .  |
+-----------------+
mawa@debian:~/workshop$ sudo ssh-keygen -t dsa -f raspberry/etc/ssh/ssh_host_dsa_key
Generating public/private dsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in raspberry/etc/ssh/ssh_host_dsa_key.
Your public key has been saved in raspberry/etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
5c:19:bc:e2:dc:d3:05:d9:63:22:04:ed:5f:76:d1:a4 root@debian
The key's randomart image is:
+---[DSA 1024]----+
|         .=o  o .o|
|           +o+ +o.|
|          .oo +E..|
|        ...o   + .|
|        oSo o + . |
|         o o o    |
|            .     |
|                  |
|                  |
+-----------------+
```

Im vorletzten Schritt muss für das Raspbian-System die Netzwerkkonfiguration mit einer festen IP-Adresse erstellt werden, damit der Raspberry später über diese erreichbar und mittels SSH konfiguriert werden kann. Hierbei muss etwas vorgegriffen der Abschnitt *„Einrichten einer statischen Netzwerkkonfiguration„* bearbeitet werden und eine feste IP-Adresse für den Raspberry festgelegt werden. Diese ist danach mit dem Werkzeug *Nano* in der Datei ***raspberry/etc/network/interfaces*** einzutragen.

```
mawa@debian:~/workshop$ sudo nano raspberry/etc/network/interfaces
```

```
auto lo

iface lo inet loopback

auto eth0
iface eth0 inet static
        address         192.168.178.201
        netmask         255.255.255.0
        gateway         192.168.178.1
        dns-nameservers 8.8.8.8
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Wenn die Datei abgespeichert wurde, muss zum Abschluss das Systemimage mit dem Werkzeug *umount* wieder aus dem Dateibaum demontiert werden.

```
mawa@debian:~/workshop$ sudo umount raspberry/
```

Ist das ohne Fehlermeldung geglückt, kann die SD-Card entfernt und in den Raspberry eingesetzt werden. Wenn dieser an das Netzwerk angeschlossen ist, kann auf diesen mittels einer SSH-Sitzung zugegriffen werden. Der hierbei zu verwendende Benutzername ist ***pi*** mit dem Passwort ***raspberry***.

```
mawa@debian:~/workshop$ ssh pi@192.168.178.201
key_load_public: invalid format
The authenticity of host '192.168.178.201 (192.168.178.201)' can't be established.
ECDSA key fingerprint is 7e:29:cf:d0:0a:26:76:3a:8c:22:e5:f5:1b:e0:b4:32.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.178.201' (ECDSA) to the list of known hosts.
pi@192.168.178.201's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi ~ $
```

Nach dem erfolgreichen Einloggen kann jetzt die Raspberry-Konfiguration manuell gestartet und abgearbeitet werden.

```
pi@seafile:~$ sudo raspi-config
```

Vorbereitung der Initialisierung mittels Maus und Tastatur

Für die Initialisierung sind ein externer Monitor und eine USB-Tastatur notwendig. Folgende Schritte sind einzuhalten:

1. HDMI zu VGA-Adapter an den Raspberry Pi anschließen.
2. VGA-Monitor an den HDMI zu VGA-ADAPTER anschließen.
3. USB-Tastatur an den Raspberry Pi anschließen.
4. Speicherkarte in den Raspberry Pi einsetzen.
5. Alle Anschlüsse kontrollieren.
6. Stromversorgung anschließen und Raspberry Pi starten.

Die Initialisierung von Raspbian und Minibian unterscheidet sich nur geringfügig.

Initialisierung von Raspbian

Nach dem ersten Start mit einer neu beschriebenen Speicherkarte startet das Raspberry Pi Konfigurationsmenü oder kann manuell aufgerufen werden.

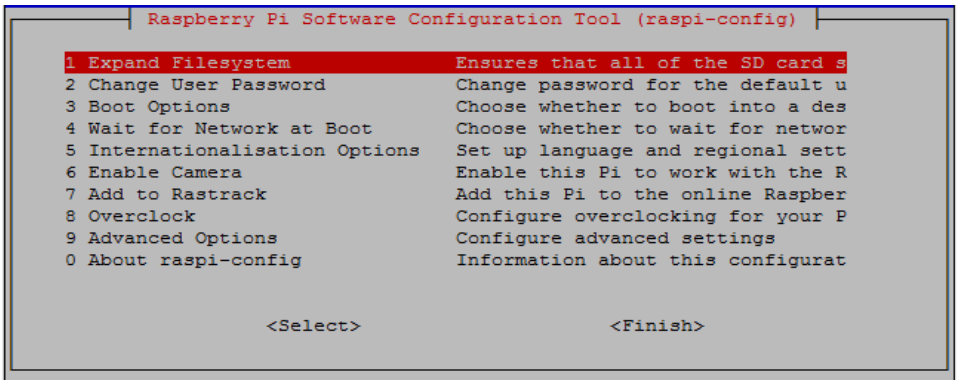

Abbildung 1: Raspberry Pi Konfigurationsmenü

Es stehen folgende Konfigurationseinstellungen zur Verfügung:

Expand Filesystem

Raspbian hat auf der SD-Speicherkarte ein 3GB große Systempartition angelegt. Bei einer 4GB Standard SD-Speicherkarte liegen, damit ca 1GB Speicherplatz brach, und werden nicht verwendet. Mit dieser Konfigurationsoption vergrößert Raspbian automatisch die Systempartition auf die maximal verfügbare Speichergröße der SD-Speicherkarte.

Die Konfigurationsoption sollte durchgeführt werden und kann eine oder zwei Minuten dauern.

Change User Password

Der Standardbenutzer bei Raspbian trägt den Benutzernamen „pi“ und hat das Standardkennwort „raspbian“. Hier kann im Vorfelde das Kennwort des Benutzers geändert werden.

Aus Sicherheitsgründen sollte diese Option durchgeführt und ein sicheres Kennwort gesetzt werden.

Boot Options

Hier wir festgelegt, wie Raspian gestartet werden soll. Zur Auswahl stehen nur Konsolenanmeldung, grafische Anmeldung und beide Option mit automatischem Login.

Da die grafische Benutzeroberfläche für den Serverbetrieb nicht benötigt wird, sollte diese Konfigurationsoption übersprungen werden.

Internationalisation Options

Hier verbergen sich drei Konfigurationsoptionen, die für die Spracheinstellung wichtig sind.

Change Locale – Auswahl des Zeichensatzes und der Sprache für das System. *Hier sollte auf DE.UTF-8 umgestellt und diese Einstellung als Standard bestätigt werden.*

Change Timezone – Einstellung der Zeitzone, in der der Raspberry Pi betrieben wird. *Da wir für den Serverbetrieb eine aktuelle Zeitzone haben wollen, sollten hier Europa und Berlin eingestellt werden.*

Change Keyboard Layout – Einstellung für das Tastaturlayout und die verwendete Sprachversion der Tastatur. *Hier sollte als Layout „Generic 105-key“ und als Sprachversion „german“ eingestellt werden.*

Change Wi-fi Country *– Einstellung des Landesschemas für die WLAN-Schnittstelle. Aus funkrechtlichen Gründen muss hier in Deutschland DE für Deutschland eingestellt sein.*

Enable Camera

Aktivieren des Treibers für die Pi-Cam.

Da nicht geplant ist die Kamera im Serverbetrieb zu betreiben, sollte diese Konfigurationsoption übersprungen werden.

Add to Rastrack

Ermöglich es die Raspbian-Installation auf Projektseite Rastrack[2] zu registrieren.

Aus Gründen der Datensparsamkeit sollte auf die Verwendung dieses Dienstes verzichtet werden.

Overclock

Diese Konfigurationsoption bietet die Möglichkeit den Raspberry Pi zu übertaktet und hier etwas mehr Rechenleistung heraus zu holen.

Aus Gründen der Systemstabilität sollte darauf verzichtet werden.

Advanced Options

Overscan – Zum Betrieb eines Monitors oder eines Fernsehers kann diese Option eingeschaltet werden. *Da der Raspberry Pi später nur über das Netzwerk oder einen HDMI-Monitor konfiguriert wird, sollte diese Option deaktiviert werden.*

Hostname – Konfiguration eines individuellen Rechnernamen für den Raspberry Pi. *Um die verschiedenen Rechner später im Netzwerk sicher auseinanderhalten zu können, sollte hier ein zur Aufgabe des Rechners passender Name gewählt werden. Z.B. Seafile.*

Memory Split – Hier kann die Aufteilung des Arbeitsspeichers nach Hauptspeicher zu Videospeicher vorgenommen werden. *Wird der Raspberry Pi ohne grafische Benutzeroberfläche betrieben, kann hier von der Standardeinstellung abgewichen und der Speicher für den Grafikprozessor auf 16MB begrenzt werden. Der restliche Speicher wird dann automatisch dem Hauptspeicher zugeordnet.*

SSH aktivieren – Aktiviert den SSH-Dienst auf dem Raspbian. Damit ist es dann möglich, den Raspberry Pi über das Netzwerk zu erreichen und zu konfigurieren. *Diese Option muss konfiguriert werden, wenn dieses nicht bereits manuell in der Vorbereitung geschehen ist.*

SPI – Bindet die SPI-Kerneltreiber mit ein und lädt diese beim Systemstart automatisch. *Da Speicher knapp ist und der SPI-Bus für den Serverbetrieb nicht benötigt wird, sollte diese Option nicht aktiviert werden.*

2 http://rastrack.co.uk/

I2C – Bindet die I2C-Kerneltreiber mit ein und lädt diese beim Systemstart automatisch. *Da Speicher knapp ist und der I2C-Bus für den Serverbetrieb nicht benötigt wird, sollte diese Option nicht aktiviert werden.*

Serial – Aktivieren einer seriellen Textkonsole über den seriellen Port des Raspberry Pi. Über diese ist es möglich, Fehlermeldungen über ein serielles Kabel zu lesen und den Rechner zu konfigurieren. *Diese Option eröffnet die Möglichkeit, im Notfall nicht erst einen Monitor und eine Tastatur zum Raspberry Pi schleppen zu müssen, sondern den Raspberry Pi einfach über ein Notebook mit seriellem Wandler konfigurieren zu können. Daher sollte diese Option aktiviert werden.*

Tipp:

Wenn es mit der Konfiguration nicht geklappt, oder man vergessen hat wichtige Einstellungen vorzunehmen, kann das Konfigurationswerkzeug auch später über die Konsole gestartet werden. Hier ist ggf. bei der Anmeldung als Benutzer „pi“ darauf zu achten, dass das Kennwort mit einem englischen Tastaturlayout eingegeben werden muss und daher „y“ und „z“ vertauscht sind und Sonderzeichen nicht dort liegen, wo man sie erwartet. Das Werkzeug für die Konfiguration ist *raspi-config*.

```
pi@seafile:~$ sudo raspi-config
```

Einrichten einer statischen Netzwerkkonfiguration

Damit der Seafile-Server später im Netzwerk gut zu erreichen ist, sollte er später direkt an den Router eingesteckt werden und über eine feste IP-Konfiguration für das Netzwerk verfügen. Wenn man die dynamische IP-Adresse seines Raspberry Pi beim Einstecken an das Netzwerk kennt, kann man diese Konfiguration bereits über einen SSH-Client mittels des Konfigurationsrechers durchführen. Da der Raspberry Pi aber nach der Initialisierung eventuell an Monitor und Tastatur hängt, ist es möglich, diese Einstellung direkt vorzunehmen.

Bevor man sich jetzt aber auf die entsprechende Konfigurationsdatei stürzt, sollte man sich ein paar Gedanken über die gewünschte IP-Konfiguration machen. Für die Konfiguration werden folgende Parameter benötigt:

- IP-Adresse des Rechners
- Netzwerkmaske des Netzwerkes
- IP-Adresse des Standardrouters
- IP-Adresse des DNS-Dienstes

Als Beispiel ist hier einmal die Standardkonfiguration der FritzBox angenommen, die im Einzugsbereich häufig vorhanden sein sollte. Ein über eine FritzBox konfiguriertes Netzwerk hat folgende Basisparameter:

- Netzwerkmaske des Netzwerkes ist **255.255.255.0**, also ein altes klassisches Klasse C-Netzwerk.
- Die IP-Adresse des Standardrouters ist die **192.168.178.1**.
- Die IP-Adresse des DNS-Dienstes entspricht der IP-Adresse des Standardrouters und lautet daher ebenfalls **192.168.178.1**
- Die IP-Adressen der Clientrechers im Netzwerk vergibt der Standardrouter mittels eines Konfigurationsdienstes dynamisch. Hierbei vergibt er Adressen aus einem voreingestellten Adresspool, der bei der FritzBox den Bereich von **192.168.178.20 bis 192.168.178.200** umfasst. Da IP-Adressen im Netzwerk eindeutig zu vergeben sind, ist es zu vermeiden, dass eine IP-Adresse mehrfach im Netzwerk vergeben wird. Daher bleibt uns bei der Wahl der festen IP-Adresse und nur eine Adresse übrig, die nicht im entsprechenden Pool liegt und auch nicht anderweitig im Netzwerk verwendet wird. Bei einer Fritzbox bieten sich daher IP-Adressen von **192.168.178.201 bis 192.168.179.254** an.

Sollte im Netzwerk eine andere Konfiguration vorliegen, so sind zuerst die entsprechenden Parameter zu ermitteln. Dabei kann es helfen, in der Routerkonfiguration nachzuschauen und

zu prüfen, welche Konfiguration ein Netzwerkclient vom Router dynamisch zugewiesen bekommt.

Für das hier verwendete Beispiel können folgende Parameter genommen werden:

- IP-Adresse: **192.168.178.201**
- Netzwerkmaske: **255.255.255.0**
- Standardrouter: **192.168.178.1**
- DNS-Server: **192.168.178.1**

Die Konfiguration der Netzwerkschnittstelle erfolgt über die Datei */etc/network/interfaces*. Hier ist mit dem Texteditor *„nano“* die Konfiguration der eth0-Schnittstelle auf statisch umzustellen und die Parameter über entsprechende Parameter zu konfigurieren.

```
pi@seafile ~ $ sudo nano /etc/network/interfaces
```

Da Systemdateien nur mit administrativen Berechtigungen geschrieben werden können, muss der Texteditor mit dem Werkzeug sudo aufgerufen werden. Die Konfigurationsdatei muss dann auf folgenden Inhalt geändert und gespeichert werden:

```
auto lo

iface lo inet loopback

auto eth0
iface eth0 inet static
        address         192.168.178.201
        netmask         255.255.255.0
        gateway         192.168.178.1
        dns-nameservers 8.8.8.8

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Nachdem auch diese Änderung durchgeführt und gespeichert wurde, ist es an der Zeit, den Raspberry Pi neu zu starten.

```
pi@seafile ~ $ sudo shutdown -r now
```

Nach dem Neustart sollte der Raspberry Pi jetzt unter seiner festen IP-Adresse im Netzwerk erreichbar sein und es sollte ein SSH-Dienst auf Verbindungsanfragen warten. Dieses wird jetzt mit einer Testverbindung vom Konfigurationscomputer und dem Werkzeug *ssh* überprüft. Die Anmeldung am Raspberry Pi soll natürlich mit dem eingerichteten Benutzer „pi“ erfolgen.

```
mawa@debian:~$ ssh pi@192.168.178.201
The authenticity of host '192.168.178.201 (192.168.178.201)' can't be established.
ECDSA key fingerprint is 20:27:8c:d9:4e:a4:98:1b:85:d0:ea:a4:d3:4a:91:b1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.178.201' (ECDSA) to the list of known hosts.
pi@192.168.178.201's password:
Linux seafile 3.12.35+ #730 PREEMPT Fri Dec 19 18:31:24 GMT 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 15 20:15:15 2015 from 192.168.178.250
pi@seafile ~ $
```

Die SSH-Verbindung ist erfolgreich. Da wir uns das erste Mal mit diesem Rechner verbinden, fragt das Werkzeug nach, ob wir diesen Rechner kennen und den Fingerabdruck des Hostschlüssels bestätigen können. Nach dem dieser bestätigt wurde, wird der Fingerabdruck in einer Konfigurationsdatei gespeichert und bei weiteren Verbindungen zum selben Rechner mit dem gleichen Fingerabdruck nicht mehr abgefragt. Für die Anmeldung als Benutzer „pi“ wird noch das Passwort eingegeben und danach steht die Kommandozeile auf dem Raspberry Pi bereit und wartet auf unsere Befehle.

Womit jetzt ein guter Zeitpunkt erreicht ist, die Internetverbindung auf dem Raspberry Pi zu testen und dem Betriebssystem ein paar Updates zu verpassen.

Update des Betriebssystems

Das schöne an Installations-Images ist, das sie sich leicht auf Datenträgern installieren lassen, aber dafür schon am nächsten Tag veraltet sind. Daher ist es jetzt an der Zeit mit dem Werkzeug *aptitude* vorhandene Betriebssystemupdates über das Internet abzufragen und zu installieren. Da auch dieses Werkzeug administrative Rechte benötigt ist es, wie alle anderen administrativen Werkzeuge, über Werkzeug *sudo* aufzurufen. Der erste Teil ruft hierbei mit der Funktion „*update“* eine aktualisierte Liste der verfügbaren Pakete ab.

```
pi@seafile ~ $ sudo aptitude update
Holen: 1 http://mirrordirector.raspbian.org wheezy Release.gpg [490 B]
Holen: 2 http://mirrordirector.raspbian.org wheezy Release [14,4 kB]
Holen: 3 http://archive.raspberrypi.org wheezy Release.gpg [490 B]
Treffer http://raspberrypi.collabora.com wheezy Release.gpg
Treffer http://raspberrypi.collabora.com wheezy Release
Holen: 4 http://archive.raspberrypi.org wheezy Release [10,2 kB]
Holen: 5 http://mirrordirector.raspbian.org wheezy/main armhf Packages [6.894 kB]
Treffer http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Holen: 6 http://archive.raspberrypi.org wheezy/main armhf Packages [106 kB]
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-de_DE
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-de
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://archive.raspberrypi.org wheezy/main Translation-de_DE
Ign http://archive.raspberrypi.org wheezy/main Translation-de
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Treffer http://mirrordirector.raspbian.org wheezy/contrib armhf Packages
Treffer http://mirrordirector.raspbian.org wheezy/non-free armhf Packages
Treffer http://mirrordirector.raspbian.org wheezy/rpi armhf Packages
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-de_DE
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-de
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-de_DE
Ign http://mirrordirector.raspbian.org wheezy/main Translation-de
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-de_DE
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-de
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-de_DE
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-de
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
7.025 kB wurden in 33 s heruntergeladen (213 kB/s)

Aktueller Status: 24 aktualisierbare Pakete [+24], 6 Neue [+6].
pi@seafile ~ $
```

Es werden die benötigten Paketdatenbanken herunter geladen und am Ende festgestellt, das für mindestens 24 Softwarepakete ein Update zur Verfügung steht.

Die vorhandenen Updates können jetzt nach dem Paketabgleich über das Internet herunter geladen und installiert werden. Hierfür biete das Werkzeug **aptitude** die Funktion *„safe-upgrade“* an, das ein Upgrade innerhalb einer Versionslinie des Betriebssystems ermöglicht.

```
pi@seafile ~ $ sudo aptitude safe-upgrade
Die folgenden Pakete werden aktualisiert:
  base-files curl dosfstools file libc-bin libc-dev-bin libc6 libc6-dev
  libcurl3 libcurl3-gnutls libevent-2.0-5 libmagic1 locales mime-support
  multiarch-support perl perl-base perl-modules python-picamera
  python-rpi.gpio python3-picamera python3-rpi.gpio tzdata unzip
24 Pakete aktualisiert, 0 zusätzlich installiert, 0 werden entfernt und 0 nicht
aktualisiert.
24,8 MB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 33,8
kB frei werden.
Möchten Sie fortsetzen? [Y/n/?] Y
Holen: 1 http://archive.raspberrypi.org/debian/ wheezy/main python-rpi.gpio armhf
0.5.9-1 [43,2 kB]
Holen: 2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main base-files armhf
7.1wheezy8+rpi1 [67,7 kB]
Holen: 3 http://archive.raspberrypi.org/debian/ wheezy/main python3-rpi.gpio armhf
0.5.9-1 [25,4 kB]

[ ... ]

24,8 MB wurden in 15 s heruntergeladen (1.648 kB/s)
Vorkonfiguration der Pakete …

[ ... ]

(Lese Datenbank ... 74923 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Ersetzen von base-files 7.1wheezy6+rpi1 (durch .../base-
files_7.1wheezy8+rpi1_armhf.deb) ...
Ersatz für base-files wird entpackt ...
Trigger für install-info werden verarbeitet ...
Trigger für man-db werden verarbeitet ...
base-files (7.1wheezy8+rpi1) wird eingerichtet ...
Neue Version der Konfigurationsdatei /etc/debian_version wird installiert ...
(Lese Datenbank ... 74923 Dateien und Verzeichnisse sind derzeit installiert.)

[ ... ]

libc-dev-bin (2.13-38+rpi2+deb7u6) wird eingerichtet ...
libc6-dev:armhf (2.13-38+rpi2+deb7u6) wird eingerichtet ...
libevent-2.0-5:armhf (2.0.19-stable-3+deb7u1) wird eingerichtet ...

[ ... ]

Aktueller Status: 0 aktualisierbare Pakete [-24].
pi@seafile ~ $
```

Der Upradeprozess kann sich aufgrund der geringen Leistungsfähigkeit des Raspberry Pi etwas in die Länge ziehen. Innerhalb dieses Prozesses werden die einzelnen Softwarepakete heruntergeladen, entpackt, in das Dateisystem installiert und eingerichtet. Am Ende dieses Prozesses sollte der aktuelle Status ergeben, dass keine weiteren Pakete mehr zu aktualisieren sind.

Konnte dieser Prozess erfolgreich durchgeführt werden, steht der Raspberry Pi über eine feste IP-Adresse im lokalen Netzwerk zur Verfügung und ist in der Lage auf das Internet zuzugreifen. Deshalb kann er jetzt heruntergefahren und ggf. vom Monitor und Tastatur getrennt werden.

```
pi@seafile ~ $ sudo shutdown -h now
```

Vorbereiten und Einbindung des Datenspeichers

Im nächsten Schritt geht es darum, den Datenspeicher vorzubereiten und fest einzubinden. Hierfür muss jetzt der USB-Speicherstick im ausgeschalteten Zustand in den Raspberry Pi eingesetzt und der Raspberry Pi danach gestartet werden. Mittels des Werkzeugs *ssh* verbindet man sich für die weiteren dmesgArbeiten mit dem Raspberry Pi über seine feste IP-Adresse.

Nach dem Neustart sollte der USB-Stick als Datenspeicher vom Kernel erkannt und es sollte einer Gerätedatei zugeordnet worden sein. Jetzt gilt es, für die weiteren Arbeiten den Namen dieser Gerätedatei heraus zu bekommen.

```
pi@seafile ~ $ dmesg

[ ... ]

[    4.609375] scsi 0:0:0:0: Direct-Access     Intenso  Basic Line       8.07 PQ:
0 ANSI: 4
[    4.636331] sd 0:0:0:0: [sda] 7680000 512-byte logical blocks: (3.93 GB/3.66
GiB)
[    4.657241] sd 0:0:0:0: [sda] Write Protect is off
[    4.682587] sd 0:0:0:0: [sda] Mode Sense: 23 00 00 00
[    4.683662] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled,
doesn't support DPO or FUA
[    4.722231]  sda: sda1
[    4.741061] sd 0:0:0:0: [sda] Attached SCSI removable disk
[    4.807713] udevd[160]: starting version 175
[   12.720096] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[   13.517717] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[   23.219872] smsc95xx 1-1.1:1.0 eth0: hardware isn't capable of remote wakeup
[   24.664269] smsc95xx 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
[   28.939964] Adding 102396k swap on /var/swap.  Priority:-1 extents:3
across:1982076k SSFS
```

Die letzen Zeilen der Logausgabe geben Aufschluss über die erkannten Datenträger und ihre Gerätebezeichnungen. Der USB-Stick wurde als Gerät *sda* erkannt und enthält eine Partition *sda1*. Um sicherzugehen, kann hier auch noch die Größe des USB-Sticks abgeglichen werden, die im Beispiel der Speichergröße von ca. 4GB entspricht. Dieses wird jetzt noch einmal mit den erkannten Partitionen des Kernels über das virtuelle Proc-Dateisystem abgeglichen, die man sich dem Werkzeug *cat* anschaut.

```
pi@seafile ~ $ cat /proc/partitions
major minor  #blocks  name

 179        0    7782400 mmcblk0
 179        1      57344 mmcblk0p1
 179        2    7720960 mmcblk0p2
   8        0    3840000 sda
   8        1    3838976 sda1
```

Die Ausgabe dieser Kernelinformation gibt uns Gewissheit, die gesuchte Partition trägt den Namen sda1. Bei den erkannten Partitionen, die mit mmcblk anfangen, handelt es sich um die Partition auf der SD-Speicherkarte, auf dem das Betriebssystem läuft.

Da jetzt die Bezeichnung der zu bearbeitenden Gerätedatei bekannt ist, ist es an der Zeit diese zu formatieren und damit ein linuxspezifisches Dateisystem auf der betreffenden Partition zu installieren. Aber vorher noch einmal eine obligatorische Warnung:

> **WARNUNG**
>
> **Beim Formatieren einer Partition gehen alle Daten auf dem beschrieben Gerät verloren. Daher prüfen Sie bitte vorher noch einmal genau, ob wirklich das richtige Gerät und die zugehöhrige Gerätedatei verwendet werden. Das Beschreiben der falschen Gerätedatei kann zu Datenverlust und Startproblemen auf dem Raspberry Pi oder dem Konfigurations-computer führen.**
>
> **Schreiben Sie *nicht* einfach die Parameter dieser Beispielinstallation ab, sondern ermitteln Sie die für Sie gültigen Parameter!**
>
> **Prüfen Sie vor dem Formatieren auch, das Sie wirklich über SSH auf dem Raspberry Pi arbeiten und nicht versehentlich eine Partition des Konfigurationscomputers löschen!**

Zum Formatieren der Partition kommt das Werkzeug *mkfs* (Make Filesystem) zum Einsatz. Dieses bekommt als Parameter den gewünschten Dateisystemtyp *ext4* und die Gerätedatei der zu formatierenden Partition. Als Systemwerkzeug ist auch dieses Werkzeug mit administrativen Rechten auszuführen, um einen wirksamen Effekt zu erzielen.

```
pi@seafile ~ $ sudo mkfs -t ext4 /dev/sda1
mke2fs 1.42.5 (29-Jul-2012)
Dateisystem-Label=
OS-Typ: Linux
Blockgröße=4096 (log=2)
Fragmentgröße=4096 (log=2)
Stride=0 Blöcke, Stripebreite=0 Blöcke
240000 Inodes, 959744 Blöcke
47987 Blöcke (5.00%) reserviert für den Superuser
Erster Datenblock=0
Maximale Dateisystem-Blöcke=985661440
30 Blockgruppen
32768 Blöcke pro Gruppe, 32768 Fragmente pro Gruppe
8000 Inodes pro Gruppe
Superblock-Sicherungskopien gespeichert in den Blöcken:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Platz für Gruppentabellen wird angefordert: erledigt
Inode-Tabellen werden geschrieben: erledigt
Erstelle Journal (16384 Blöcke): erledigt
Schreibe Superblöcke und Dateisystem-Accountinginformationen: erledigt
pi@seafile ~ $
```

Der Vorgang kann je nach Größe des USB-Sticks mehrere Minuten in Anspruch nehmen und es werden verschiedene Informationen über das initialisierte Dateisystem ausgegeben. Sollte hierbei keine Fehler auftreten, ist die neue Partition für den Einsatz unter Linux bereit.

Jetzt ist es an der Zeit im Dateisystem des installierten Betriebssystems einen Montagepunkt zu erstellen, an den das neue Dateisystem vom USB-Stick montiert werden kann. Hierbei ist wichtig zu wissen, das Linux keine Laufwerksbuchstaben kennt und Dateisysteme von Laufwerken und Wechseldatenträgern fest in die Hierarchie des Wurzeldateisystems einbindet. Damit dieses funktioniert, wird jetzt ein neuer Montagepunkt erstellt. Als Montagepunkte dienen einfache Verzeichnisse, daher wird einfach mit dem Werkzeug *mkdir* ein neues Verzeichnis unterhalb des Verzeichnisses */srv* erstellt. Das Verzeichnis */srv* dient als Ablage für Serverdienste und da passt unser neues Verzeichnis mit der Bezeichnung *seafile* thematisch wunderbar hinein.

```
pi@seafile ~ $ sudo mkdir /srv/seafile
```

Zu beachten ist hier noch, dass das Linux-Dateisystem Groß- und Kleinschreibung unterscheidet. Daher kann es hilfreich sein, Datei- und Verzeichnisnamen einheitlich in Kleinschreibung zu wählen.

Jetzt kommt ein schwieriger Punkt. Es gilt, die Partition des USB-Sticks automatisch beim Systemstart am Montagepunkt zu montieren. Dabei muss man wissen, dass nicht sichergestellt ist, dass die Partition beim nächsten Start seinen Gerätenamen behält. Die Zuordnung der Gerätenamen erfolgt zum Startzeitpunkt über die Reihenfolge in denen die Geräte von den Treibern erkannt und zugeordnet werden. Diese kann davon abhängig sein, ob weitere Geräte angeschlossen werden oder ob das Gerät in einen anderem Anschluss eingesteckt wurde. Daher können wir die Zuordnung nicht einfach vom Gerätenamen abhängig machen, sondern sollten hierfür eine eindeutige ID verwenden, die beim Einrichten der Partition in diese geschrieben wurde. Dieses *UUID* (*Universally Unique Identifier*) sollte ein Gerät eindeutig identifizieren und mit dem Werkzeug blkid bekommen wir alle UUIDs der vom Betriebssystem erkannten Blockgeräte angezeigt.

```
pi@seafile ~ $ sudo blkid
/dev/mmcblk0p1: SEC_TYPE="msdos" LABEL="boot" UUID="936C-7154" TYPE="vfat"
/dev/mmcblk0p2: UUID="c1398422-7a7c-4863-8a8f-45a1db26b4f2" TYPE="ext4"
/dev/sda1: UUID="d2ac472c-83d3-4617-bd71-a261e17fcd3c" TYPE="ext4"
```

In dieser Ausgabe ist auch unsere Partition *sda1* aufgeführt, die ein ext4-Dateisystem enthält. Die UUID dieser Partition lautet: ***d2ac472c-83d3-4617-bd71-a261e17fcd3c*** und ist eindeutig. Das bedeutet, sie kann eindeutig über diese ID identifiziert werden. Richten wir einen weiteren USB-Stick als Datenspeicher ein, wird die neu eingerichtete Partion eine andere UUID enthalten. Damit haben wir die Möglichkeit, eine Partition mit einer gewünschten UUID fest mit einem bestimmten Montagepunkt zu koppeln. Wollen wir später diese Partition gegen eine andere Partition austauschen, um z.B. mehr Daten speichern zu können, müssen an der entsprechenden Stelle in einer Konfigurationsdatei die alte UUID gegen die neue UUID ausgetauscht werden. Da sonst die neue Partition nicht montiert wird und alle neuen Daten in das Wurzeldateisystem geschrieben werden.

Die entsprechende Konfigurationsdatei, die für die automatische Montage von Dateisystemen und Partitionen zuständig ist, ist die Datei */etc/fstab*. Diese muss jetzt eine weitere Konfigurationszeile erhalten, die unsere Partition über die UUID mit dem Montagepunkt */srv/seafile* verbindet. Dafür wird die Datei über den Texteditor *nano* bearbeitet.

```
pi@seafile ~ $ sudo nano /etc/fstab
```

Hierbei ist folgende Änderung in der letzten Zeile hinzuzufügen und zu speichern:

```
proc                                      /proc         proc  defaults             0  0
/dev/mmcblk0p1                            /boot         vfat  defaults             0  2
/dev/mmcblk0p2                            /             ext4  defaults,noatime     0  1
# a swapfile is not a swap partition, so no using swapon|off from here on, use  dphys-
swapfile swap[on|off]  for that

UUID=d2ac472c-83d3-4617-bd71-a261e17fcd3c  /srv/seafile  ext4  defaults,noatime,rw,sync
0 2
```

Der Aufbau der Konfigurationsdatei entspricht einer Tabelle, deren einzelne Spalten durch einen Tabulator oder ein Leerzeichen getrennt sind. Die einzelnen Spalten stehen in unserem Beispiel für Folgendes:

UUID=d2ac472c-83d3-4617-bd71-a261e17fcd3c

Dieses ist die UUID der zu montierenden Partition, nur wenn diese passt, wird das Dateisystem am Montagepunkt montiert.

/srv/seafile

Das ist der gewählte Montagepunkt, an dem die Partition montiert werden soll.

ext4

Das Dateisystem, das auf der Partition genutzt wird und das für die Montage verwendet werden soll. Passt diese Angabe nicht zur Partition, wird die Partition nicht montiert.

defaults,noatime,rw

Hierbei handelt es sich um Optionen, die den Montageprozess steuern. Die Partition wird mit den *Standardparametern* (defaults) und *Schreibzugriff* (rw) montiert. Zusätzlich wird beim Lesen von Dateien darauf verzichtet, die *Zugriffszeit* (noatime) in das Dateisystem zu schreiben. Das reduziert die Schreibzugriffe auf das Medium, was sich bei Flashspeichern wie USB-Sticks positiv auf die Lebensdauer auswirkt. Was aber bei Systemen mit geringer Stabilität und empfindlichen Wechseldatenträgern vor Datenverlust schützen kann. Bei all diesen Optionen ist es wichtig, dass es sich um eine kommaseparierte Liste handelt, deren einzelne Optionen nur mit einem Komma zu trennen sind.

Nach einem Neustart des Raspberry Pi kann jetzt mit dem Werkzeug *df* (disk free) geprüft werden, welche Partitionen wo in das Wurzeldateisystem montiert wurden.

```
pi@seafile ~ $ sudo shutdown -h now
```

```
pi@seafile ~ $ df
Dateisystem    1K-Blöcke Benutzt Verfügbar Verw% Eingehängt auf
rootfs           7534284 2477636   4700332   35% /
/dev/root        7534284 2477636   4700332   35% /
devtmpfs          244128       0    244128    0% /dev
tmpfs              49660     244     49416    1% /run
tmpfs               5120       0      5120    0% /run/lock
tmpfs              99300       0     99300    0% /run/shm
/dev/mmcblk0p1     57288    9920     47368   18% /boot
/dev/sda1        3713136    7512   3497292    1% /srv/seafile
```

Wenn alles richtig funktioniert hat, erscheint jetzt in der Liste unsere Partition vom USB-Stick am richtigen Montagepunkt. Hier können wir auch sehen, wie viel Speicherplatz auf den einzelnen montierten Laufwerken noch zur Verfügung steht.

Installation und Konfiguration des Seafile Servers

Wenn die Betriebssystembasis steht, ist es soweit, dass der Serverdienst von Seafile installiert werden kann. In diesem Workshop wird die Installation des Seafile Servers in der Version 5.x für eine private Umgebung durchgeführt.

Die Installation des Seafile-Servers erfolgt unterhalb des eingerichteten Verzeichnisses */srv/seafile*. Damit wir dort als Benutzer „pi“ möglichst ungestört arbeiten können, ändern wir den Eigentümer des Verzeichnisses mit dem Werkzeug *chown* auf den Benutzer „pi“ und wechseln danach in dieses Verzeichnis.

```
pi@seafile ~ $ sudo chown pi:pi /srv/seafile/
pi@seafile ~ $ ls -l -a /srv/seafile
insgesamt 24
drwxr-xr-x 3 pi   pi    4096 Jan 16 18:30 .
drwxr-xr-x 3 root root  4096 Jan 16 18:57 ..
drwx------ 2 root root 16384 Jan 16 18:30 lost+found
pi@seafile ~ $ cd /srv/seafile
pi@seafile /srv/seafile $ pwd
/srv/seafile
```

Nach dem änder des Eigentümers für das Verzeichnis, werden die neuen Besitzverhältnisse mit dem Werkzeug *ls* geprüft. Hierbei sind die Optionen -l (liste) -a (all) wichtig, um die Attribute und die versteckten Dateien und Verzeichnisse angezeigt zu bekommen. Ist alles korrekt, wird mit dem Werkzeug *cd* (change directory) in das Serververzeichnis gewechselt und das neue Arbeitsverzeichnis überprüft. Alle weiteren Arbeiten werden jetzt aus diesem Verzeichnis heraus durchgeführt, sofern es in den weiteren Schritten nicht anders angegeben ist.

Die weitere Installation lehnt sich an die Installation des Seafile Servers aus dem entsprechenden Wiki[1] an. An einigen Stellen kann es aber zu Abweichungen gegenüber der Wikireferenz kommen.

1 http://manual.seafile.com/deploy/using_sqlite.html

Installieren der Verzeichnisstruktur und der Serverdateien

In diesem Schritt muss das Installationspaket geladen und die Datei und Verzeichnisstruktur aufgebaut werden. Das Herunterladen kann wieder bequem mit dem Werkzeug wget erfolgen, den dafür benötigten Downloadlink kopiert man sich einfach von der Seafile-Downloadseite[2]. Hierbei muss darauf geachtet werden, den aktuellen Link für den Server auf Raspberry Pi zu verwenden.

```
pi@raspberrypi:/srv/seafile $ wget https://download.seafile.de/seafile-
server_latest-stable_jobenvil_pi.tar.gz
--2016-04-08 22:12:14--  https://download.seafile.de/seafile-server_latest-
stable_jobenvil_pi.tar.gz
Auflösen des Hostnamen »download.seafile.de (download.seafile.de)«...
148.251.100.148
Verbindungsaufbau zu download.seafile.de (download.seafile.de)|
148.251.100.148|:443... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 302 Moved Temporarily
Platz: https://github.com/haiwen/seafile-rpi/releases/download/v5.0.5/seafile-
server_stable_jobenvil_5.0.5_pi.tar.gz[folge]
--2016-04-08 22:12:14--  https://github.com/haiwen/seafile-
rpi/releases/download/v5.0.5/seafile-server_stable_jobenvil_5.0.5_pi.tar.gz
Auflösen des Hostnamen »github.com (github.com)«... 192.30.252.120
Verbindungsaufbau zu github.com (github.com)|192.30.252.120|:443... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 302 Found
Platz: https://github-cloud.s3.amazonaws.com/releases/34115988/4fbb4cea-e172-11e5-
8fc4-72e53abca807.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAISTNZFOVBIJMK3TQ%2F20160408%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20160408T201215Z&X-Amz-Expires=300&X-Amz-
Signature=2186e9c293c04b3df02fbc9f06e413495ee7fd81efb44544f9494c4e581003cc&X-Amz-
SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B
%20filename%3Dseafile-server_stable_jobenvil_5.0.5_pi.tar.gz&response-content-
type=application%2Foctet-stream[folge]
--2016-04-08 22:12:15--  https://github-
cloud.s3.amazonaws.com/releases/34115988/4fbb4cea-e172-11e5-8fc4-72e53abca807.gz?
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAISTNZFOVBIJMK3TQ
%2F20160408%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20160408T201215Z&X-Amz-
Expires=300&X-Amz-
Signature=2186e9c293c04b3df02fbc9f06e413495ee7fd81efb44544f9494c4e581003cc&X-Amz-
SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B
%20filename%3Dseafile-server_stable_jobenvil_5.0.5_pi.tar.gz&response-content-
type=application%2Foctet-stream
Auflösen des Hostnamen »github-cloud.s3.amazonaws.com (github-
cloud.s3.amazonaws.com)«... 54.231.49.138
Verbindungsaufbau zu github-cloud.s3.amazonaws.com (github-
cloud.s3.amazonaws.com)|54.231.49.138|:443... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 21775921 (21M) [application/octet-stream]
In »»seafile-server_latest-stable_jobenvil_pi.tar.gz«« speichern.

seafile-server_latest 100%[=======================>]  20,77M  3,72MB/s   in 7,0s

2016-04-08 22:12:22 (2,96 MB/s) - »»seafile-server_latest-
stable_jobenvil_pi.tar.gz«« gespeichert [21775921/21775921]

pi@raspberrypi:/srv/seafile $
```

Nach dem Herunterladen des Serverpaketes muss dieses im Serverordner entpackt werden, was mit dem Werkzeug *tar* durchgeführt wird. Danach ist ein Ablageverzeichnis mit dem Namen *installed* zu erstellen und das Installationspaket für die spätere Verwendung in dieses Verzeichnis mit dem Werkzeug *mv* (move) zu verschieben und zu archiviert.

2 http://seafile.com/en/download/

```
pi@raspberrypi:/srv/seafile $ tar -xzf seafile-server_latest-
stable_jobenvil_pi.tar.gz
pi@raspberrypi:/srv/seafile $ mkdir installed
pi@raspberrypi:/srv/seafile $ mv seafile-server_latest-stable_jobenvil_pi.tar.gz
installed/
```

Das Archiv stellt sicher, das wir später bei Bedarf noch Zugriff auf die Installationsdateien einer bestimmten Version haben. Was sich später sich beim Zurückspielen von Backupständen oder bei Upgrades als nützlich erweisen könnte.

Installation benötigter Softwarepakete

Damit der Server installiert werden kann, werden zusätzliche Softwarepakete aus dem Raspbian Repository benötigt, die mit dem Werkzeug aptitude zu laden und installieren sind.

```
pi@seafile /srv/seafile $ sudo aptitude install python2.7 python-setuptools
python-imaging sqlite3
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  python-imaging python-pkg-resources{a} python-setuptools sqlite3
0 Pakete aktualisiert, 4 zusätzlich installiert, 0 werden entfernt und 0 nicht
aktualisiert.
1.043 kB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden
2.708 kB zusätzlich belegt sein.
Möchten Sie fortsetzen? [Y/n/?]
Holen: 1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main python-imaging
armhf 1.1.7-4+deb7u1 [413 kB]
Holen: 2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main python-pkg-
resources all 0.6.24-1 [63,6 kB]
Holen: 3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main python-
setuptools all 0.6.24-1 [449 kB]
Holen: 4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main sqlite3 armhf
3.7.13-1+deb7u1 [118 kB]
1.043 kB wurden in 1 s heruntergeladen (889 kB/s)
Vormals nicht ausgewähltes Paket python-imaging wird gewählt.
(Lese Datenbank ... 74927 Dateien und Verzeichnisse sind derzeit installiert.)
Entpacken von python-imaging (aus .../python-imaging_1.1.7-4+deb7u1_armhf.deb) ...
Vormals nicht ausgewähltes Paket python-pkg-resources wird gewählt.
Entpacken von python-pkg-resources (aus .../python-pkg-resources_0.6.24-1_all.deb)
...
Vormals nicht ausgewähltes Paket python-setuptools wird gewählt.
Entpacken von python-setuptools (aus .../python-setuptools_0.6.24-1_all.deb) ...
Vormals nicht ausgewähltes Paket sqlite3 wird gewählt.
Entpacken von sqlite3 (aus .../sqlite3_3.7.13-1+deb7u1_armhf.deb) ...
Trigger für man-db werden verarbeitet ...
python-imaging (1.1.7-4+deb7u1) wird eingerichtet ...
python-pkg-resources (0.6.24-1) wird eingerichtet ...
python-setuptools (0.6.24-1) wird eingerichtet ...
sqlite3 (3.7.13-1+deb7u1) wird eingerichtet ...
```

Wurden die Softwarepakte korrekt installiert, steht dem Setup des Seafile-Servers nichts mehr im Wege.

Setup des Seafile-Servers

Setup gestaltet sich auf den ersten Blick relativ einfach, man wechselt einfach in das Serververzeichnis und startet dort das gewünschte Setup-Script.

```
ppi@raspberrypi:/srv/seafile $ cd seafile-server-5.0.5/
pi@raspberrypi:/srv/seafile/seafile-server-5.0.5 $ ./setup-seafile.sh
-----------------------------------------------------------------
This script will guide you to config and setup your seafile server.

Make sure you have read seafile server manual at

        https://github.com/haiwen/seafile/wiki

Note: This script will guide your to setup seafile server using sqlite3,
which may have problems if your disk is on a NFS/CIFS/USB.
In these cases, we sugguest you setup seafile server using MySQL.

Press [ENTER] to continue
-----------------------------------------------------------------


Checking packages needed by seafile ...

Checking python on this machine ...
Find python: python2.7

  Checking python module: setuptools ... Done.
  Checking python module: python-imaging ... Done.
  Checking python module: python-sqlite3 ... Done.

Checking for sqlite3 ...Done.

Checking Done.

What would you like to use as the name of this seafile server?
Your seafile users will be able to see the name in their seafile client.
You can use a-z, A-Z, 0-9, _ and -, and the length should be 3 ~ 15
[server name]: mySeafile

What is the ip or domain of this server?
For example, www.mycompany.com, or, 192.168.1.101

[This server's ip or domain]: 127.0.0.1
```

Im ersten Abschnitt des Konfigurationsscripts werden Softwareabhängigkeiten und die sqlite Datenbank auf Verfügbarkeit und Version geprüft. Während des Installationsprozesses wird man ggf. dazu aufgefordert, die Enter-Taster für den weiteren Verlauf zu drücken oder im zweiten Abschnitt einige wichtige Parameter zu konfigurieren.

Der erste Parameter ist hierbei der Name des Servers, der später in den Seafileclients als Servername angezeigt werden soll. Im Beispiel wurde hier der Name *mySeafile* gewählt.

Der zweite Parameter ist die IP-Adresse, unter der der Server zu erreichen sein wird. Da wir später mit einem SSH-Tunnel darauf zugreifen wollen, laute die IP-Adresse hier *127.0.0.1*, was später Auswirkungen hat, wenn Dateien über die Webschnittstelle heruntergeladen werden sollen. Dieser Parameter kann später bei Bedarf in der Konfigurationsdatei manuell geändert werden.

Die weiteren Konfigurationsparameter sind bei der Sqlite-Version relativ harmlos. Es geht hierbei um den Serverport, der für den Dateiserver verwendet werden soll und um den Pfad für die Datenablage. Hier können mit der Enter-Taste die Standardwerte bestätigt werden.

```
Where would you like to store your seafile data?
Note: Please use a volume with enough free space.
[default: /srv/seafile/seafile-data ]

What tcp port do you want to use for seafile fileserver?
8082 is the recommended port.
[default: 8082 ]


This is your config information:

server name:        mySeafile
server ip/domain:   127.0.0.1
seafile data dir:   /srv/seafile/seafile-data
fileserver port:    8082
```

Am Ende gibt es noch eine Konfigurationsübersicht, aus der auch das Datenverzeichnis entnommen werden kann. Diese Parameter sollte man sich für die spätere Verwendung ggf. sichern oder ausdrucken.

Nach beherztem Betätigen der Enter-Taste, läuft der weitere Installationsvorgang ohne weitere Unterbrechungen durch.

In diesem letzten Abschnitt werden Dienste und Datenbanken eingerichtet.


```
Generating ccnet configuration in /srv/seafile/ccnet...

done
Successly create configuration dir /srv/seafile/ccnet.

Generating seafile configuration in /srv/seafile/seafile-data ...

Done.

-----------------------------------------------------------------
Seahub is the web interface for seafile server.
Now let's setup seahub configuration. Press [ENTER] to continue
-----------------------------------------------------------------


Creating seahub database now, it may take one minute, please wait...


Done.

creating seafile-server-latest symbolic link ... done


-----------------------------------------------------------------
Your seafile server configuration has been completed successfully.
-----------------------------------------------------------------

run seafile server:     ./seafile.sh { start | stop | restart }
run seahub  server:     ./seahub.sh  { start <port> | stop | restart <port> }

-----------------------------------------------------------------
If the server is behind a firewall, remember to open these tcp ports:
-----------------------------------------------------------------

port of seafile fileserver:   8082
port of seahub:               8000

When problems occur, refer to

      https://github.com/haiwen/seafile/wiki

for more information.
```

Am Ende des Vorganges werden noch wichtige Informationen ausgegeben, wie der Seafiledienst manuell gestartet und beendet werden kann. Auch die wichtigen Ports, die für den externen Zugriff in der Firewall geöffnet sein müssen, werden hier noch einmal explizit benannt. Diese Angaben sind für den späteren Verlauf auch für unsere Installation wichtig, auch wenn wir diese Ports nicht direkt in das Internet zur Verfügung stellen wollen.

Zum Abschluss der Installation müssen jetzt die Dienste *Seafile* und *Seahub* einmal manuell gestartet werden. Dieses ist wichtig, weil beim ersten Start des Seahub-Dienstes der Seafileadministrator und sein Kennwort festgelegt werden müssen.

```
pi@seafile /srv/seafile/seafile-server-4.3.2 $ ./seafile.sh start

Starting seafile server, please wait ...
Seafile server started

Done.
pi@seafile /srv/seafile/seafile-server-4.3.2 $ ./seahub.sh start

Starting seahub at port 8000 ...

----------------------------------------
It's the first time you start the seafile server. Now let's create the admin
account
----------------------------------------

What is the email for the admin account?
[ admin email ] admin@nomail.loc

What is the password for the admin account?
[ admin password ]

Enter the password again:
[ admin password again ]



----------------------------------------
Successfully created seafile admin
----------------------------------------



Seahub is started

Done.
```

Nach dem manuellen Start der Dienste kann der Zugriff über den Seafilehub mit einem Browser getestet werden. Auch eine Anmeldung mit gesetzten Administratordaten ist möglich.

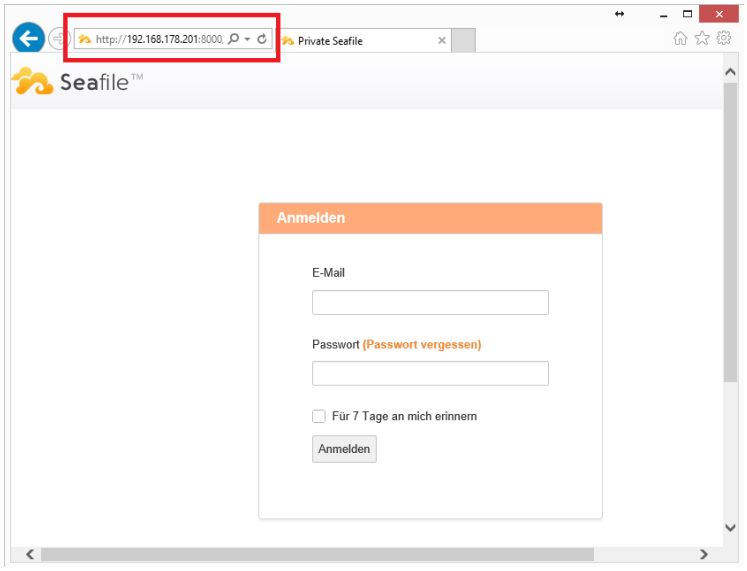

Abbildung 2: Erster Browserzugriff auf den Seafileserver

Das automatische Starten der Seafile-Dienste einrichten

Zurzeit müssen die Dienste immer manuell gestartet oder gestoppt werden, was für einen möglichst autonomen Serverbetrieb hinderlich ist. Daher ist es jetzt an der Zeit, diesen Vorgang in den Startvorgang des Betriebssystems einzubinden. Hierfür müssen zuerst die Dienste wieder manuell beendet werden. Danach wechseln wir wieder in das Seafile-Grundverzeichnis.

```
pi@raspberrypi:/srv/seafile/seafile-server-5.0.5 $ ./seahub.sh stop

Stopping seahub ...
Done.

pi@raspberrypi:/srv/seafile/seafile-server-5.0.5 $ ./seahub.sh stop

Seahub is not running
Done.

pi@raspberrypi:/srv/seafile/seafile-server-5.0.5 $ cd /srv/seafile/
pi@raspberrypi:/srv/seafile $
```

Anlegen eines Seafile Benutzers für die Serverdienste

Da die Dienste später nicht mit administrativen Rechten laufen sollen, muss zuerst ein neuer Benutzer angelegt werden, mit deren Rechten die Dienste ausgeführt werden. Das Anlegen neuer Benutzer im System erfolgt mit dem Werkzeug *adduser*, das hierfür administrative Rechte benötigt.

```
pi@seafile /srv/seafile $ sudo adduser --home /srv/seafile --no-create-home
--disabled-login seafile
Lege Benutzer »seafile« an ...
Lege neue Gruppe »seafile« (1004) an ...
Lege neuen Benutzer »seafile« (1001) mit Gruppe »seafile« an ...
Erstelle Home-Verzeichnis »/srv/seafile« nicht.
Benutzerinformationen für seafile werden geändert.
Geben Sie einen neuen Wert an oder drücken Sie ENTER für den Standardwert
        Vollständiger Name []:
        Zimmernummer []:
        Telefon geschäftlich []:
        Telefon privat []:
        Sonstiges []:
Sind die Informationen korrekt? [J/n]
pi@seafile /srv/seafile $
```

Dem Werkzeug werden verschieden Optionen und Parameter mitgegeben, die folgende Aufgaben erfüllen:

--home

Legt das Heimatverzeichnis des Benutzers fest. In diesem Falle bekommt der Benutzer das Seafile-Grundverzeichnis als Heimatverzeichnis zugewiesen.

--no-create-home

Verhindert, dass ein neues Heimatverzeichnis erstellt wird. Das ist auch nicht notwendig, da dieses bereits erstellt und mit Leben gefüllt wurde.

--disabled-login
: Kennzeichnet den Benutzer als inaktiv und verhindert, dass man sich mit diesem Konto über eine Konsole am System anmelden kann.

seafile
: Name des neuen Benutzers.

Beim Anlegevorgang wird man nach weiteren informativen Benutzerinformationen gefragt, die man mit der Enter-Taste einfach leer bestätigt, da diese Informationen für unseren Dienstbenutzer nicht benötigt werden.

Abstraktion des Seafile-Programmverzeichnisses

Damit später ein einfacheres Upgrade der Serverdienste möglich ist, ist es sinnvoll, das Programmverzeichnis des Seafile-Servers mittels eines symbolischen Links zu abstrahieren. Damit ist es später möglich, mehrere Binärinstallationen nebeneinander im Grundverzeichnis zu haben und den symbolischen Link auf die jeweils aktuelle Installation zu setzten. Das erspart spätere Anpassung an den noch zu erstellenden Startskripten. Ein Blick in das Grundverzeichnis offenbart, das die Seafileinstallation uns diese Arbeit schon abgenommen hat:

```
pi@seafile ln -s seafile-server-latest seafile-server-5.0.5/
pi@seafile /srv/seafile $ ls -l
insgesamt 188
drwx------ 6 pi   pi     4096 Jan 16 22:58 ccnet
drwxr-xr-x 2 pi   pi     4096 Jan 16 22:23 conf
drwxr-xr-x 2 pi   pi     4096 Jan 16 21:50 installed
drwxr-xr-x 2 pi   pi     4096 Jan 16 23:01 logs
drwx------ 2 root root  16384 Jan 16 18:30 lost+found
drwxr-xr-x 2 pi   pi     4096 Jan 16 23:27 pids
drwxr-xr-x 8 pi   pi     4096 Jan 16 22:58 seafile-data
drwxr-xr-x 6 pi   pi     4096 Dez 16 04:22 seafile-server-4.0.1
lrwxrwxrwx 1 pi   pi       20 Jan 16 22:24 seafile-server-latest -> seafile-
server-4.0.1
drwxr-xr-x 3 pi   pi     4096 Jan 16 22:24 seahub-data
-rw-r--r-- 1 pi   pi   128000 Jan 16 23:11 seahub.db
-rw-r--r-- 1 pi   pi       40 Jan 16 22:23 seahub_settings.py
-rw-r--r-- 1 pi   pi      212 Jan 16 23:01 seahub_settings.pyc
```

Der symbolische Link *seafile-server-latest* verweist hierbei auf die letzte aktuelle Seafile-Installation.

Eigentumsverhältnisse der Dateien übertragen

Damit der eingerichtete Seafile-Benutzer die Möglichkeit hat Daten und Dateien abzulegen, wird er jetzt mit dem Werkzeug *chown* zum Eigentümer des Seafile-Grundverzeichnisses, alle Unterverzeichnisse und Dateien gemacht. Damit dieser Vorgang rekursiv die gesamte Verzeichnisstruktur durchläuft, wird dem Werkzeug die Option *-R* übergeben.

```
pi@seafile /srv/seafile $ sudo chown seafile:seafile /srv/seafile/ -R
```

Erstellen eines Startscripts

Zum Starten der Dienste wird ein Startscript benötigt. Zum Glück stellt das Seafile Handbuch bereits entsprechende Vorlagen zur Verfügung[3], die ggf. noch ein paar kleinere Anpassungen bedürfen und unten als Vorlage abgedruckt ist. Das Erstellen der Datei kann mittels des Texteditors *nano* erfolgen, der einfach eine neue Datei /etc/init.d/seafile-server erstellt.

```
pi@seafile /srv/seafile $ sudo nano /etc/init.d/seafile-server
```

In diese Datei kann der Inhalt der unteren Vorlage einkopiert und die betreffenden Konfigurationsvariablen bei Bedarf angepasst werden.

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          seafile-server
# Required-Start:    $local_fs $remote_fs $network
# Required-Stop:     $local_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Starts Seafile Server
# Description:       starts Seafile Server
### END INIT INFO

# Change the value of "user" to your linux user name
user=seafile

# Change the value of "script_path" to your path of seafile installation
seafile_dir=/srv/seafile
script_path=${seafile_dir}/seafile-server-latest
seafile_init_log=${seafile_dir}/logs/seafile.init.log
seahub_init_log=${seafile_dir}/logs/seahub.init.log

# Change the value of fastcgi to true if fastcgi is to be used
fastcgi=false
# Set the port of fastcgi, default is 8000. Change it if you need different.
fastcgi_port=8000

case "$1" in
        start)
                sudo -u ${user} ${script_path}/seafile.sh start >> ${seafile_init_log}
                if [  $fastcgi = true ];
                then
                        sudo -u ${user} ${script_path}/seahub.sh start-fastcgi ${fastcgi_port}
>> ${seahub_init_log}
                else
                        sudo -u ${user} ${script_path}/seahub.sh start >> ${seahub_init_log}
                fi
        ;;
        restart)
                sudo -u ${user} ${script_path}/seafile.sh restart >> ${seafile_init_log}
                if [  $fastcgi = true ];
                then
                        sudo -u ${user} ${script_path}/seahub.sh restart-fastcgi $
{fastcgi_port} >> ${seahub_init_log}
                else
                        sudo -u ${user} ${script_path}/seahub.sh restart >> ${seahub_init_log}
                fi
        ;;
        stop)
                sudo -u ${user} ${script_path}/seafile.sh $1 >> ${seafile_init_log}
                sudo -u ${user} ${script_path}/seahub.sh $1 >> ${seahub_init_log}
        ;;
        *)
                echo "Usage: /etc/init.d/seafile {start|stop|restart}"
                exit 1
        ;;
esac
```

3 http://manual.seafile.com/deploy/start_Seafile_at_system_bootup.html

Als wichtige Parameter sollte man ggf. folgende Umgebungsvariablen im Script anpassen:

user=seafile

Dieses ist der Benutzername, unter dem die Dienste laufen sollen. Er sollte dem extra dafür angelegtem Benutzer entsprechen.

seafile_dir=/srv/seafile

Das Grundverzeichnis, unter dem die Seafile-Dienste installiert sind.

fastcgi=false

Legt fest, ob fastcgi für den Zugriff über einen Webserver verwendet werden sollen. Da dieser Dienst zurzeit noch nicht installiert ist, sollte diese Funktion jetzt noch deaktiviert werden.

fastcgi_port=8000

Der verwendete Port für den FastCGI-Zugriff.

Nach dem Anlegen und Bearbeiten des Startscripts muss dieses noch über die Anpassung der Dateiberechtigungen das Recht zum Ausführen, also zum Starten gegeben werden. Zusätzlich muss das Script in den Startprozess eingebunden werden.

Die Berechtigungen der Datei werden mit dem Werkzeug chmod angepasst und das Einbinden in den Startablauf erfolgt mittels des Werkzeugs update-rc.d. Beide Werkzeuge benötigen administrative Rechte für ihre Aufgaben.

```
pi@seafile /srv/seafile $ sudo chmod +x /etc/init.d/seafile-server
pi@seafile /srv/seafile $ sudo update-rc.d seafile-server defaults
update-rc.d: using dependency based boot sequencing
```

Danach kann der Raspberry Pi neu gestartet und er Start der Seafile-Dienste getestet werden. Hierbei ist zu beachten, dass der Start gerne ein paar Minuten Zeit in Anspruch nehmen kann, da der Raspberry Pi leider keinen V8 unter der Haube hat.

Seahub über Nginx verfügbar machen

Normalerweise ist die Installation jetzt bereit für den Arbeitseinsatz. Allerdings muss der Hub dann immer über einen ungünstigen Port aufgerufen werden und das Webinterface kommt sehr behebig daher. Daher macht es Sinn, einen Reverse-Proxy mit FastCGI-Unterstützung einzusetzen. Hierfür bietet sich der Webserver Nginx als leichtgewichtige Lösung an.

Die Einrichtung und Konfiguration entspricht weitestgehend dem Seaflle Installationshandbuch[4].

Installation von Nginx

Zuerst müssen der Webserver, das entsprechende Pythonmodul und abhängige Softwarepakete aus dem Repository heruntergeladen und installiert werden. Dieses wird über das Werkzeug *aptitude* erledigt.

```
pi@seafile ~ $ sudo aptitude install nginx python-flup
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  geoip-database{a} libgeoip1{a} libpq5{a} nginx nginx-common{a}
  nginx-full{a} python-cheetah{a} python-egenix-mxdatetime{a}
  python-egenix-mxtools{a} python-flup python-psycopg2{a} python-webpy{a}
  python2.6{a} python2.6-minimal{a}
0 Pakete aktualisiert, 14 zusätzlich installiert, 0 werden entfernt und 0 nicht
aktualisiert.
6.905 kB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 22
,9 MB zusätzlich belegt sein.
Möchten Sie fortsetzen? [Y/n/?]
Holen: 1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main geoip-database
all 20130213-1 [1.466 kB]

[ ... ]

6.905 kB wurden in 5 s heruntergeladen (1.185 kB/s)
Vormals nicht ausgewähltes Paket geoip-database wird gewählt.
(Lese Datenbank ... 75411 Dateien und Verzeichnisse sind derzeit installiert.)
Entpacken von geoip-database (aus .../geoip-database_20130213-1_all.deb) …

[ ... ]
python-webpy (1:0.37+20120626-1) wird eingerichtet ...
Trigger für python-support werden verarbeitet ...
Trigger für menu werden verarbeitet  ...
pi@seafile ~ $
```

Für das weitere Vorgehen sind jetzt die Seafile-Dienste zu beenden, damit die entsprechenden Konfigurationen vorgenommen werden können. Da bereits im vorherigen Abschnitt ein Startscript für die Dienste erstellt wurde, kommt hier zum Beenden der Dienste das administrative Werkzeug *service* zum Einsatz.

```
pi@seafile ~ $ sudo service seafile-server stop
```


4 http://manual.seafile.com/deploy/deploy_with_nginx.html

Konfiguration von Nginx

Im nächsten Schritt ist die Konfigurationsdatei vom Nginx anzupassen. Das Seafile Installationshandbuch hält hierfür schon eine gute Vorlage bereit, die auch die Konfiguration von FastCGI beinhaltet. Diese muss nur noch an unsere Bedürfnisse angepasst und in die richtige Konfigurationsdatei geschrieben werden. Dieses ist die Konfigurationsdatei */etc/nginx/sites-available/default*, die mit dem Texteditor *nano* bearbeitet werden muss.

```
pi@seafile ~ $ sudo nano /etc/nginx/sites-available/default
```

Deren Inhalt ist wie folgt anzupassen:

```
server {
    listen 80;
    server_name 127.0.0.1;
    location / {
        fastcgi_pass    127.0.0.1:8000;
        fastcgi_param   SCRIPT_FILENAME     $document_root$fastcgi_script_name;
        fastcgi_param   PATH_INFO           $fastcgi_script_name;

        fastcgi_param   SERVER_PROTOCOL     $server_protocol;
        fastcgi_param   QUERY_STRING        $query_string;
        fastcgi_param   REQUEST_METHOD      $request_method;
        fastcgi_param   CONTENT_TYPE        $content_type;
        fastcgi_param   CONTENT_LENGTH      $content_length;
        fastcgi_param   SERVER_ADDR         $server_addr;
        fastcgi_param   SERVER_PORT         $server_port;
        fastcgi_param   SERVER_NAME         $server_name;
        fastcgi_param   REMOTE_ADDR         $remote_addr;

        access_log      /var/log/nginx/seahub.access.log;
        error_log       /var/log/nginx/seahub.error.log;
    }

    location /seafhttp {
        rewrite ^/seafhttp(.*)$ $1 break;
        proxy_pass http://127.0.0.1:8082;
        client_max_body_size 0;
        proxy_connect_timeout  36000s;
        proxy_read_timeout  36000s;
    }

    location /media {
        root /srv/seafile/seafile-server-latest/seahub;
    }
}
```

Anpassen der Seafile Dienste

An einigen Konfigurationsdateien sind ggf. Anpassungen vorzunehmen, diese werden jetzt hier Kurzdarstellung abgebildet.

Bei der *SERVICE-URL* für den ccnet-Dienst muss der Port entfernt werden, da der Hub ab sofort über Port 80 des Webservers angesprochen werden soll.

```
pi@seafile ~ $ sudo nano /srv/seafile/ccnet/ccnet.conf
```

```
[General]
USER_NAME = mySeafile
ID = bbc60597ccdbbd02dc32741193e93b4412f21265
NAME = mySeafile
SERVICE_URL = http://127.0.0.1

[Network]
PORT = 10001

[Client]
PORT = 13418
```

Danach muss die Konfigurationsdatei des Seahub-Dienstes bearbeitet werden.

```
pi@seafile ~ $ sudo nano /srv/seafile/seahub_settings.py
```

Hier ist die Variable FILE_SERVER_ROOT hinzuzufügen und mit der richtigen URL für den Webzugriff zu füllen. Der Wert dieser URL ergibt sich ggf. aus der Konfigurationsdatei des Nginx, wo der entsprechende Pfad eingerichtet wurde.

```
SECRET_KEY = [ ... ]
FILE_SERVER_ROOT = 'http://127.0.0.1/seafhttp'
```

ANMERKUNG

Seafile wird alle Up- und Downloadoperationen immer mit der in FILE_SERVER_ROOT festgelegten URL referenzieren. Daher muss hier der gültige DNS-Name eingetragen werden, über den der Server erreicht werden kann. Sollte der Dienst später extern über einen dynamische Internetzugang zugänglich sein, muss hier die entsprechende DynDNS-Adresse eingetragen werden. Die IP-Adresse 127.0.0.1 gilt nur, wenn auf den Server mittels einem SSH-Tunnel zugegriffen wird.

Als Letztes muss jetzt noch die Startdatei angepasst werden.

```
pi@seafile ~ $ sudo nano /etc/init.d/seafile-server
```

Hier ist die Variable für FastCGI zu aktivieren.

```
[ ... ]

# Change the value of fastcgi to true if fastcgi is to be used
fastcgi=true
# Set the port of fastcgi, default is 8000. Change it if you need different.
fastcgi_port=8000

[ ... ]
```

Zum Abschluss dieses Arbeitsschrittes müssen der Webserver und die Seafile-Dienste neu gestartet werden. Hierfür kommt wieder das *service* Werkzeug zum Einsatz.

```
pi@seafile ~ $ sudo service nginx restart
Restarting nginx: nginx.
pi@seafile ~ $ sudo service seafile-server start
```

Wenn alle Einstellungen geklappt haben und die Dienste korrekt gestartet wurden, lässt sich jetzt mit einem Browser das Webinterface von Seafile direkt aufrufen.

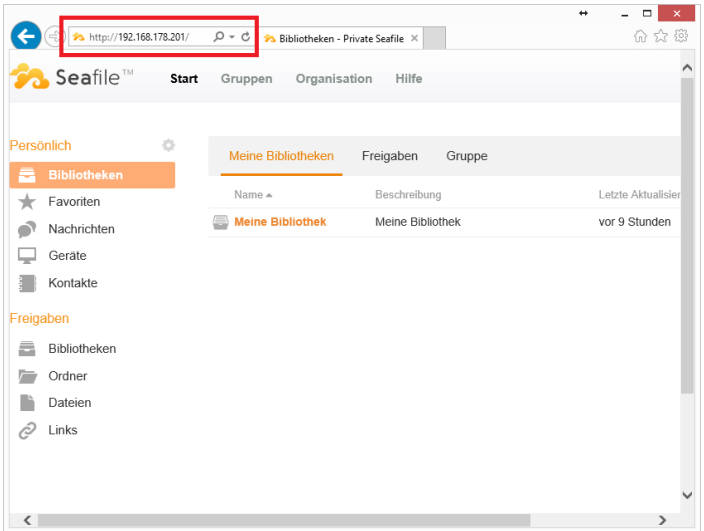

Abbildung 3: Seafile Webinterface über HTTP aufgerufen

Umstellen von Nginx auf HTTPS-Zugriff

Ist geplant, den Seafile-Dienst extern direkt ohne Tunnel oder VPN über das Internet zur Verfügung zu stellen, ist zumindest eine SSL/TLS-Verschlüsselung obligatorisch. Hierfür kommt im einfachsten Falle ein selbst signiertes Zertifikat zum Einsatz. Für das zuerst mit dem Werkzeug *openssl* ein Schlüssel mit einer Schlüssellänge von mindestens 2048 Bit zu erstellen ist. Hierbei werden alle Schlüssel und Zertifikate im Konfigurationsverzeichnis des *nginx* abgelegt.

```
pi@seafile ~ $ sudo openssl genrsa -out /etc/nginx/privkey.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................................
.............................+++
........+++
e is 65537 (0x10001)
```

Im zweiten Schritt wird mit dem Werkzeug *openssl* und dem erstellten Schlüssel ein selbst signiertes x509 Zertifikat ausgestellt. Das Zertifikat bekommt hierbei eine Laufzeit von drei Jahren (1095 Tagen) und während der Erstellung werden die Attribute für das Zertifikat abgefragt.

```
pi@seafile ~ $ sudo openssl req -new -x509 -key /etc/nginx/privkey.pem -out
/etc/nginx/cacert.pem -days 1095
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Schleswig-Holstein
Locality Name (eg, city) []:Kiel
Organization Name (eg, company) [Internet Widgits Pty Ltd]:eMWe-Datentechnik
Organizational Unit Name (eg, section) []:Seafile
Common Name (e.g. server FQDN or YOUR name) []:seatest.emwe-datentechnik.de
Email Address []:admin@nomail.loc
```

Bei den Attributen sollten sinnvolle Angaben gemacht werden, die auf den Verwendungszweck und den Aussteller schließen lassen. Beim Attribut Common-Name sollte die URL angegeben werden, über der der Seafileserver später im Netz erreichbar ist. Handelt es sich hierbei um dynamische DNS-Adresse, ist diese hier einzutragen.

Im nächsten Schritt ist die Konfiguration des Webservers Nginx auf HTTPS umzustellen. Hierfür muss die bereits vorbereitete Konfigurationsdatei */etc/nginx/sites-available/default* mit dem Texteditor *nano* angepasst werden.

```
pi@seafile ~ $ sudo nano /etc/nginx/sites-available/default
```

Bei den Änderungen wird die bisherige HTTP-Konfiguration auf HTTPS umgestellt, in dem hier der Serverport von 80 auf 443 umgestellt wird und für den klassischen HTTP-Teil ein permanentes Überschreiben (*Rewrite*) der URL auf HTTPS erfolgt. Zusätzlich werden das zu verwendende Zertifikat und der private Schlüssel festgelegt.

```
server {
    listen 80;
    server_name seatest.emwe-datentechnik.de;
    rewrite ^ https://$http_host$request_uri? permanent;
}


server {
    listen 443;
    server_name seatest.emwe-datentechnik.de;
    ssl on;
    ssl_certificate /etc/nginx/cacert.pem;
    ssl_certificate_key /etc/nginx/privkey.pem;

    location / {
        fastcgi_pass    127.0.0.1:8000;
        fastcgi_param   SCRIPT_FILENAME     $document_root$fastcgi_script_name;
        fastcgi_param   PATH_INFO           $fastcgi_script_name;

        fastcgi_param   SERVER_PROTOCOL     $server_protocol;
        fastcgi_param   QUERY_STRING        $query_string;
        fastcgi_param   REQUEST_METHOD      $request_method;
        fastcgi_param   CONTENT_TYPE        $content_type;
        fastcgi_param   CONTENT_LENGTH      $content_length;
        fastcgi_param   SERVER_ADDR         $server_addr;
        fastcgi_param   SERVER_PORT         $server_port;
        fastcgi_param   SERVER_NAME         $server_name;
        fastcgi_param   REMOTE_ADDR         $remote_addr;

        access_log      /var/log/nginx/seahub.access.log;
        error_log       /var/log/nginx/seahub.error.log;
    }

    location /seafhttp {
        rewrite ^/seafhttp(.*)$ $1 break;
        proxy_pass http://127.0.0.1:8082;
        client_max_body_size 0;
        proxy_connect_timeout  36000s;
        proxy_read_timeout  36000s;
    }

    location /media {
        root /srv/seafile/seafile-server-latest/seahub;
    }
}
```

Zusätzlich sollte hier als Server Name bereits die später zu verwendete URL eingetragen werden, auch wenn es sich hierbei um eine dynDNS-Adresse handelt.

Portweiterleitung auf dem heimischen Router konfigurieren

Soll der Seafile Server direkt aus dem Internet erreichbar sein, muss auf dem heimischen Router eine Portweiterleitung der verwendeten Dienstports eingestellt werden. Je nach gewünschter Betriebsart kommen hierfür mehrere Umleitungen infrage, die am Beispiel einer Fritzbox erklärt werden. Hierbei kann es sein, dass diese Funktionalität nicht in allen Fritzboxen vorhanden zu zugänglich ist. Bei Routern anderer Hersteller ist die Konfiguration entsprechend der Herstellerangabe durchzuführen. Die sind ggf. folgende Portweiterleitungen notwendig:

TCP-Port 22	auf Seafile-Server TCP-Port 22	Bei Nutzung v. Tunnelverbindungen.
TCP-Port 80	auf Seafile-Server TCP-Port 80	Bei HTTP/HTTPS-Zugriff.
TCP-Port 443	auf Seafile Server TCP-Port 443	Bei HTTP/HTTPS-Zugriff.

Bei der Fritzbox befinden sich die Portweiterleitungen unter dem Menüpunkt „*Freigaben*“ und dem Reiter „*Portfreigaben*“.

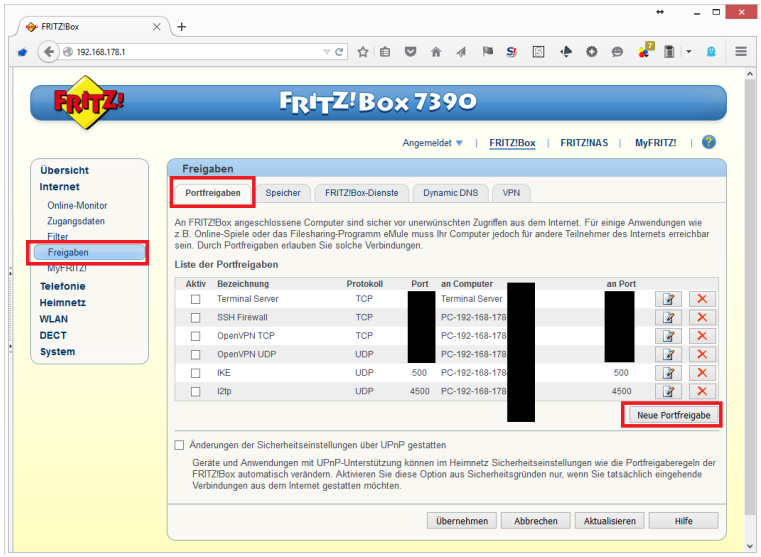

Abbildung 4: Portfreigaben auf der Fritzbox

Hier sind für alle notwendigen Freigaben neue Weiterleitungen auf den Seafile-Server einzurichten.

Bei der Einrichtung der Portfreigaben sollten alle Einstellungen und die Angabe der Ziel IP-Adresse manuell erfolgen.

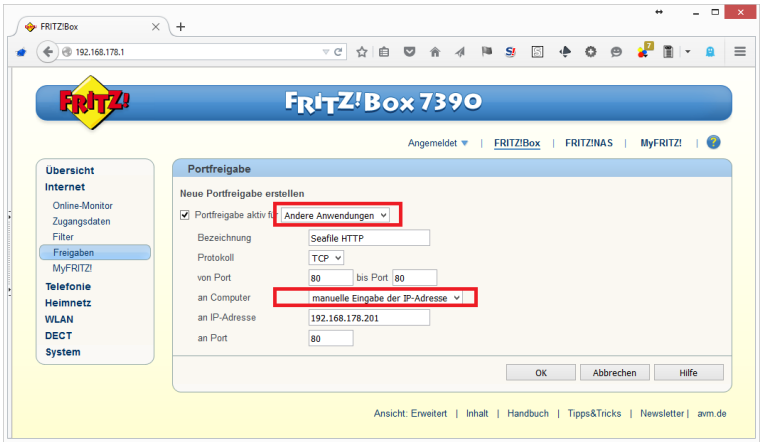

Abbildung 5: Portweiterleitung einrichten

Nachdem alle notwendigen Portweiterleitungen eingetragen wurden, müssen diese durch den Button „*Übernehmen*“ bestätigt und übernommen werden.

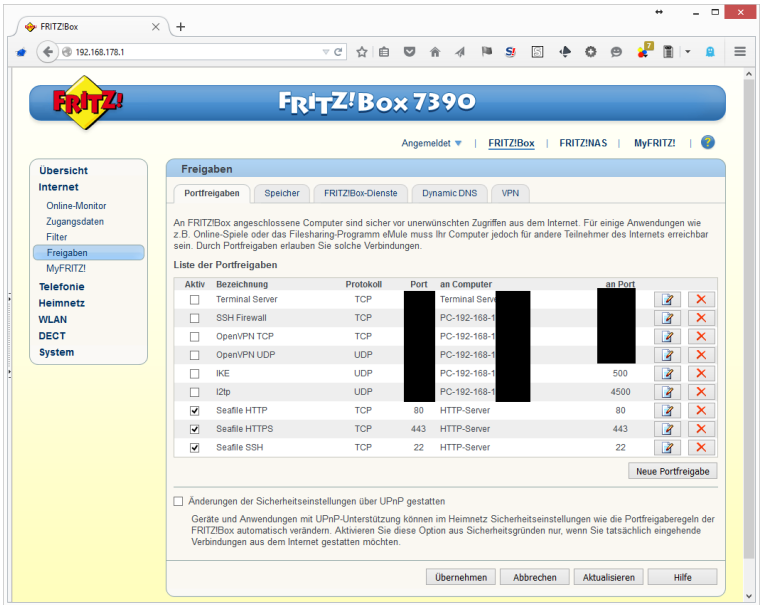

Abbildung 6: Eingerichtete Portweiterleitungen

WARNUNG

Spätestens jetzt wo der Raspberry Pi aus dem Internet erreichbar ist, muss für den Benutzer Pi ein sicheres Passwort gesetzt werden.

```
pi@seafile ~ $ passwd
Ändern des Passworts für pi.
(aktuelles) UNIX-Passwort:
Geben Sie ein neues UNIX-Passwort ein:
Geben Sie das neue UNIX-Passwort erneut ein:
passwd: Passwort erfolgreich geändert
```

Einrichten eines dynamischen DNS-Eintrags auf der Fritzbox

Damit man später seinen Seafile-Server bei wechselnder IP-Adresse auch in den Weiten des Internets wiederfindet, muss ein dynDNS-Provider eingerichtet werden, der unsere dynamische IP-Adresse auf einem festen DNS-Namen abbildet. Bei der Fritzbox ist diese Einstellung über den Menüpunkt „*Freigaben*“ und den Reiter „*Dynamisches DNS*“ verfügbar. Die Fritzbox hält hierbei auch eine Auswahl der bekanntesten Anbieter bereit, bei denen man sich jetzt einen dynamischen DNS-Namen für seine Router-IP zulegen kann.

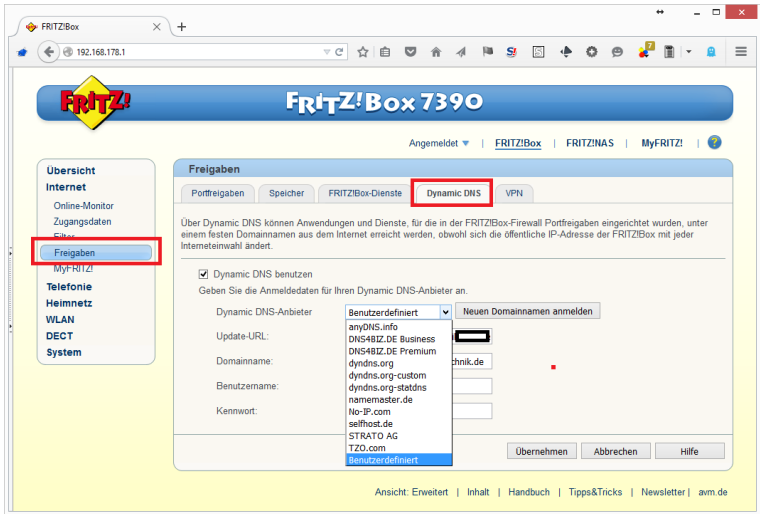

Abbildung 7: Dynamisches DNS bei der Fritzbox

Der gewählte Name sollte sowohl in dem SSL-Zertifikat, der Nginx Konfiguration und der Seafile Fileserverkonfiguration eingetragen werden.

Tunnel unter Linux aufbauen

Der einfachste Weg ist Verbindungsaufbau mihile des Werkzeugs *ssh*.

```
seatest@debian:~$ ssh connector@192.168.178.201 -N -L 8443:127.0.0.1:443 -L
8000:127.0.0.1:80
Enter passphrase for key '/home/seatest/.ssh/id_rsa':
```

Beim Aufruf des Werkzeugs verhindert die Option *-N* den Aufruf der Login Shell. Ohne diese Option würde der Server die Verbindung sofort wieder beenden. Die Optionen -L stellen die Portweiterleitung her. Hier werden als Attribut jeweils der lokale Quellport, die IP-Adresse oder der Hostname des Ziels und der Zielport erwartet. Nach der Eingabe der korrekten Passphrase des privaten Schlüssels steht die Verbindung, bis sie durch ein <STRG> C unterbrochen wird.

Anhang

Dokumente

http://s5.marek-walther.de/20160409hfd/handout.pdf	
	handout.pdf

Konfigurationsdateien

http://s5.marek-walther.de/20160409hfd/config/etc/init.d/seafile-server	
Erstellen eines Startscripts Seite: 35	/etc/init.d/seafile-server

http://s5.marek-walther.de/20160409hfd/config/etc/nginx/sites-available/default.http	
Konfiguration von Nginx Seite: 38	/etc/nginx/sites-available/default

http://s5.marek-walther.de/20160409hfd/config/etc/nginx/sites-available/default.https	
Umstellen von Nginx auf HTTPS-Zugriff Seite: 41	/etc/nginx/sites-available/default