DIETMAR **W**ALKER - **P**C-**B**LITZHELFER-**N**OTDIENST
Nationalgasse 14 • **7**2124 **P**liezhausen • ∅ **T**el. 07127 / 890 729 - **F**ax 89118
Internet: http://www.pc-blitzhelfer.de – **M**obil 0172-882 79 55

Anleitung TrueNAS

Genug gequasselt, kommen wir zum How-To.

Die Bilder stammen nicht von meiner "echten" FreeNAS 8, sondern von der Parallels-VM mit einer ähnlichen Konfiguration wie meine echte FreeNAS-Box, nur dass die Datenplatten andere Größen haben und dass die VM nur 1 GB RAM hat.
Es sei hier erwähnt: 1 GB RAM ist keine gute Idee für die Nutzung von ZFS in FreeNAS 8. ZFS will und braucht mehr. Viel mehr.

Und noch ein Tipp: Wenn das WebUI anfängt zu zicken, kann es helfen, den Browsercache zu leeren.

Die engl. Installationsanleitung mit tw. mehr Informationen, als ich sie zur Verfügung stellen werde, kann hier abgerufen werden.

1. Installation von FreeNAS 8

1.1. Installation von CD

Am einfachsten ist es, sollte ein CD-Laufwerk am Zielrechner zur Verfügung stehen, die ISO zu verwenden. Ob *i386* oder *amd64* kommt natürlich auf die zukünftige FreeNAS-Box an - falls 64 bit supported ist, besser *amd64*!
Die ISO einfach auf CD brennen, den Zielrechner von der CD starten und anschließend mittels des Installers auf ein dediziertes Medium installieren. Das Medium kann eine interne Platte sein oder ein mind. 1GB USB-Stick sein.
Vorsicht: Was auch immer als Zielmedium verwendet wird: Es wird die das gesamte Medium verwendet werden! Bei FreeNAS 7 konnte ein Teil der Zielmediums als Datenplatte verwendet werden, doch das ist bei FreeNAS 8 anders. Die Verwendung eines USB-Sticks (oder in meinem Fall einer kleinen internen SSD) ist also ratsam.
Wie die Installation von CD funktioniert ist in der englischen Anleitung gut beschrieben und bebildert.

1.2. Verwendung des xz-Images

Ist kein CD-Laufwerk an der FreeNAS-Box verfügbar, so gibt es noch die Möglichkeit, direkt am Mac die FreeNAS 8-Image auf ein USB-Stick zu kopieren. Meine FreeNAS-Box hat kein CD-LW, also war das mein Weg. Das funktioniert wie folgt:

1.2.1a. Mit MacPorts oder Homebrew (setzt Xcode voraus)
Zunächst (die) *xz*(-utils), der Einfachheit halber mittels MacPorts oder Homebrew installieren:
Mit MacPorts:
Code:

```
sudo port install xz
```

Mit Homebrew:
Code:

```
brew install xz
```

1.2.1b. Mit Keka das xz-Image extrahieren
Keka kann xz-Archive extrahieren.

1.2.2. Anschließend das FreeNAS 8 xz-Imgage herunterladen. Benötigt wird die entsprechende "*Full_Install.xz"-Datei. Auch hier gilt: ob *i386* oder *amd64* kommt natürlich auf die zukünftige FreeNAS-Box an.
Hier können die Images heruntergeladen werden.

1.2.3. Bevor das xz-Image auf den Stick kommt, ist es ratsam sich zu vergewissern, dass die Shasum stimmt. Dazu werden folgende Informationen aus den Release Notes, benötigt:
Code:

```
Filename:
FreeNAS-8.0-RELEASE-amd64.Full_Install.xz
SHA256 Hash:
e572ff58e661403786586a4a08e5031a8ada680c287cf45ed3a12a5d693147a9

Filename:
FreeNAS-8.0-RELEASE-i386.Full_Install.xz
SHA256 Hash:
3f2296a924d34da56b720720806015307c9cf6ff4d649036f23b0961dc0de21a
```

Zum Überprüfen der Shasum folgendermaßen im Terminal vorgehen:
Code:

```
shasum -a 256 /pfad/zu/FreeNAS-8.0-RELEASE-amd64.Full_Install.xz
```

Sowas sollte dabei herauskommen:
Code:

```
$> shasum -a 256 ~/Downloads/FreeNAS-8.0-RELEASE-amd64.Full_Install.xz
e572ff58e661403786586a4a08e5031a8ada680c287cf45ed3a12a5d693147a9
/Users/fyysh/Downloads/FreeNAS-8.0-RELEASE-amd64.Full_Install.xz
```

Tipp: Den Pfad des xz-Images bekommt man am einfachsten in das Terminal, wenn die Datei einfach per Drag&Drop in das Terminalfenster gezogen wird.
Stimmt der Hash? Dann auf zum nächsten Schritt.
Stimmt er nicht, dann vielleicht einen anderen Mirror verwenden und nochmal herunterladen.

1.2.4. Nun ein Terminal-Fenster öffnen (*/Programme/Diensprogramme/Terminal.app*) und folgenden Befehl eingeben:
Code:

```
diskutil list
```

Es müsste dann sowas ausgegeben werden:
Code:

```
$> diskutil list
/dev/disk0
   #:                       TYPE NAME                    SIZE
IDENTIFIER
   0:      GUID_partition_scheme                        *320.1 GB   disk0
   1:                        EFI                         209.7 MB   disk0s1
   2:                  Apple_HFS Perseus-HD              319.6 GB   disk0s2
/dev/disk1
   #:                       TYPE NAME                    SIZE
IDENTIFIER
   0:     FDisk_partition_scheme                        *16.0 GB    disk1
   1:                 DOS_FAT_32 CRUZER                     16.0 GB
disk1s1
```

Hier sieht man:
/dev/disk0 ist meine Systemplatte
/dev/disk1 ist ein bereits eingesteckter USB-Stick namens *CRUZER*

1.2.5. Jetzt wird der USB-Stick für FreeNAS 8 eingesteckt und erneut *diskutil list* ausgeführt.
Oha, da ist eine neue Disk:
Code:
```
$> diskutil list
/dev/disk0
   #:                       TYPE NAME                    SIZE
IDENTIFIER
   0:      GUID_partition_scheme                        *320.1 GB   disk0
   1:                        EFI                         209.7 MB   disk0s1
   2:                  Apple_HFS Perseus-HD              319.6 GB   disk0s2
/dev/disk1
   #:                       TYPE NAME                    SIZE
IDENTIFIER
   0:     FDisk_partition_scheme                        *2.0 GB     disk1
   1:                 DOS_FAT_32 CRUZER                      2.0 GB
disk1s1
/dev/disk2
   #:                       TYPE NAME                    SIZE
IDENTIFIER
   0:     FDisk_partition_scheme                        *2.0 GB     disk1
   1:                 DOS_FAT_32 USB-POMMES              2.0 GB     disk1s1
```
In diesem Beispiel ist also der Stick für FreeNAS 8 an */dev/disk2*. Diesem Stick hatte ich irgendwann den Namen *USB-POMMES* verliehen (er ist gelb 😉).

1.2.6. Es ist Zeit, das xz-Image mittels *xzcat* und *dd* auf den Stick zu bringen.
Und hier brauchen wir eine dicke fette rote Warnung:
ACHTUNG!
Beachte Schritt 1.2.4. & 1.2.5.!
Was auf immer im nächsten Befehl bei *dd of=/dev/disk#* steht: danach ist das, was an der dort angehängten Disk war, WEG, FORT, AUF NIMMA WIEDERSEHN!
In 99% der Fälle wird # NICHT "0" (=> NICHT */dev/disk0* verwenden!) sein! Da ist das System!
Think twice!

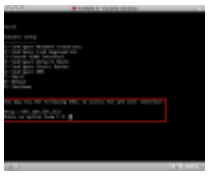
Verstanden? Super, dann weiter im Text.
Der USB-Stick muss zunächst mit folgendem Befehl unmounted (nicht ejected/ausgeworfen) werden:
Code:
```
diskutil umountDisk /dev/disk2
```
Sowas muss dabei rauskommen:
Code:
```
$> diskutil umountDisk /dev/disk2
Unmount of all volumes on disk2 was successful
```
Jetzt wird mittels folgenden Befehls das (xz-)Image auf den Stick gebracht.
Wenn 1.2.1a. befolgt wurde, d.h. xzcat auf dem Mac verfügbar ist, dann geht es so:
Code:
```
xzcat /pfad/zu/FreeNAS-8.0-RELEASE-amd64.Full_Install.xz | dd of=/dev/disk2
bs=5k
```

Wurde 1.2.1b. befolgt und das xz-Image mit Keka extrahiert, dann lautet der Befehl folgendermaßen:
Code:
```
dd if=/pfad/zu/FreeNAS-8.0-RELEASE-amd64.Full_Install of=/dev/disk2 bs=5k
```
Das dauert einige Minuten. Wenn der Vorgang fertig ist, wird im Terminal folgendes zu sehen sein:
Code:
```
122070+122071 records in
122070+122071 records out
1000000000 bytes transferred in 604.303943 secs (1654796 bytes/sec)
```
Desweiteren wird eine Fehlermeldung erscheinen, dass das Gerät nicht gemountet werden kann (oder so ähnlich) mit der Auswahl "Abbrechen", "Initiieren" und "Auswerfen" (oder so ähnlich). "Auswerfen" bzw. "Eject" klicken.
In meinem Fall musste ich das xz-Image auf die Transcend-SSD bringen, welche ich nicht einfach am Mac anschließen konnte. Deshalb habe ich ein Live-Linux von USB gebootet und habe das ganze über *SSH* gemacht - geht auch.

1.2.7. Der USB-Stick mit FreeNAS 8 kommt jetzt an die FreeNAS-Box. Von USB-Booten (das Mainboard muss das natürlich unterstützen) und falls alles gut gegangen ist, wird kurze Zeit später das FreeNAS-Menü auf dem Monitor der FreeNAS-Box angezeigt.

2. Netzwerk Konfiguration von FreeNAS 8

FreeNAS 8 konfiguriert, im Gegensatz zu FreeNAS 7, die gefundenen Netzwerkkarten (NICs) automatisch mit DHCP. Ist also einen DHCP-Server im Netzwerk verfügbar (jeder Standard-Router heutzutage kann als DHCP-Server dienen), bekommt die FreeNAS-Box automatisch eine IP und kann direkt im Brower (=> 3.) konfiguriert werden. Die IP kann in der Konsole der FreeNAS abgelesen werden:



Theoretisch ermöglicht das das hochfahren einer FreeNAS 8 von einem USB-Stick an einem PC ohne Tastatur und Monitor und anschließender Konfiguration über das WebUI. Das hätte ich mir in der Vergangenheit bei FreeNAS 7 auch gewünscht; es hätte umständliches organisieren von Monitoren und Tastaturen überflüssig gemacht 😉.

Hat die FreeNAS keine IP bekommen, dann kann es direkt an der Box konfiguriert werden. Hier am Beispiel mit der IP 192.168.0.5:

3. Grundkonfiguration von FreeNAS 8

3.1. Die in der FreeNAS-Konsole angezeigte Adresse in einem Browser öffnen und mit dem Benutzer *admin* und Passwort *freenas* anmelden:

3.2. Settings-Tab klicken und die Einstellungen anpassen:

Wichtig: Wenn auf HTTPS umstellt wird, muss die Seite über https://IP_DER_NAS erneut aufgerufen werden!

3.3. Folgendes ist nur notwendig, wenn die IP manuell konfiguriert wurde:

Ich konfiguriere die IP meiner FreeNAS immer manuell und füge der DNS-Serverlist gerne die OpenDNS-Server hinzu, so dass, falls mein Router (eine FritzBox) ein DNS-Problem hat (kommt extrem selten vor - aber kommt vor), meine FreeNAS dennoch externe IPs auflösen kann.
Aber Vorsicht: Wenn diese Einstellung gespeichert wird, dann wird sie auch komplett angewandt. D.h. es kann nicht einfach nur der Hostname und die Domain geändert werden, auch nicht, wenn DHCP eingeschaltet ist. Wird Hostname/Domain geändert und die Gateway/Nameserver Felder sind leer, dann gibt es auch keine Route mehr ins Internet und Namen können nicht aufgelöst werden! Wenn also hier etwas umgestellt wird, gleich alles richtig einstellen (oder FreeNAS neu starten - dann müsste es auch wieder gehen)!

3.4. Jetzt noch die Passwörter für den admin (WebUI) und den root (per default auch "freenas") ändern:

3.5. SSH einschalten:

Die Optionen "Login as Root with password" sowie "Allow Password Authentication" verwende ich nur zur Konfiguration (ich bin zugegebenermaßen bequem). Wenn die FreeNAS fertig eingerichtet ist mit Key-Auth etc. schalte ich diese beiden Optionen wieder aus.

3.6 Und Konfiguration sichern:

4. Migration der Daten der alten FreeNAS 7 zur FreeNAS 8

Wer dies hier nicht braucht: Einfach drüberlesen und gucken, wie man die Volumes/Zpools hinzufügt (=> 4.5., 4.9 und 4.11 sind von Interesse 😉).
Für die Kommandozeilen-Unerfahrenen: **NICHT OHNE BACKUPS**!
Für erfahrene CLI-Hacker: Trotzdem besser mit Backups! Mal nicht aufgepasst und schwups Daten weg.... ihr wisst schon. 😉

Ich hatte auf meiner alten FreeNAS 7 die gleich großen Platten (2x 2TB, 2x1TB, 2x500GB) Mittels RAID 1 jeweils zu Spiegeln zusammengefasst. Bevor ich mit der Migration der Daten begonnen habe, habe ich zur Sicherheit noch mit FreeNAS 7 die ganzen Daten auf externe USB 2.0-Festplatten gezogen. Das hat zwar Ewigkeiten gedauert, die Daten sollten für folgende Spielerei auf jeden Fall doppelt vorhanden sein! Einmal vertippt oder nicht aufgepasst und die Daten könnten weg sein!

Die anfängliche Überlegung war folgende: Auf meinem 2TB-Mirror waren ca. 60% belegt (also ca. 1,2 TB), ich brauchte also intern genau so viel Platz. Wenn ich ein Stripe (RAID 0) aus einer 1TB- und einer 500GB-Platte erstellen würde, könnte ich die Daten der 2TB Platten runterholen, diese mittels *dd* und *gmirror clear* säubern, dann den ersten ZFS-Mirror samt Pool erstellen, die Daten aus dem Stripe wieder zurück auf den 2TB-ZFS-Mirror übertragen usw. und das eben mit den internen SATA-Geschwindigkeiten statt den wesentlich geringeren USB-2.0 Geschwindigkeiten.

Da ich mir aufgrund der Erfahrungen mit FreeNAS 8 RC5 nicht sicher war, dass mein Vorhaben über das WebUI funktionieren würde, in der RC5 hätte es auf keinen Fall funktioniert, habe ich fast ausschließlich das CLI (Command Line Interface) verwendet und nur die endgültigen ZFS-Mirrors/Pools im WebUI erstellt. Es hat perfekt funktioniert.

Das Vorgehen habe ich für dieses How-To mit der oben angesprochenen VM nachgestellt.

Hierzu noch ein Hinweis:
FreeNAS 8 erstellt auf jedem neuen ZFS-*vdev* eine Swap-Partition. Die Größe dieser Partition lässt sich über "Settings -> Advanced" einstellen. Der Standardwert ist "2", ein kleinerer Wert als "1" ist nicht möglich.
Wen ein Zpool erstellt werden soll, jedoch nicht genügend Speicherplatz für die Swap-Partition

vorhanden ist, schlägt die Erstellung ohne Nennung des Grundes fehl. Bei "normalen" Plattengrößen ist das weitestgehend egal, da für gewöhnlich immer genug Platz vorhanden sein wird, testet man jedoch in einer VM, so sollte man dies im Hinterkopf haben. Die Mindestgröße für eine Platte muss also 2GB sein, von denen dann nur 1GB zur Verfügung stehen wird.
Es gibt einen Weg, um das zu umgehen: den Zpool im CLI erstellen, exportieren und dann im WebUI importieren. In einer VM kein Problem.
Dies ist jedoch nicht die empfohlene Vorgehensweise mit echten Festplatten, denn das Erstellen von ZFS-vdevs und -Pools über das WebUI macht mehr, als nur ein Pool zu erstellen und eine Swap-Partition darauf zu klastchen: es wendet auch Tricks an, um Festplatten mit 4k-Sektoren richtig zu handeln und das wiederum kommt der ZFS-Performance zugute.

4.1. Zunächst musste ich herausfinden, welche Platte welche wo angehängt ist:
Code:
```
$> ssh root@192.168.254.223
root@192.168.254.223's password:
[...]
Welcome to my fancy FreeNAS 8
freenas# dmesg | grep -e "ada[0-9]:"
[...]
ada0: 2048MB (4194304 512 byte sectors: 16H 63S/T 4161C)
[...]
ada1: 5120MB (10485760 512 byte sectors: 16H 63S/T 10402C)
[...]
ada2: 5120MB (10485760 512 byte sectors: 16H 63S/T 10402C)
[...]
ada3: 10240MB (20971520 512 byte sectors: 16H 32S/T 16383C)
[...]
ada4: 10240MB (20971520 512 byte sectors: 16H 32S/T 16383C)
[...]
ada5: 20480MB (41943040 512 byte sectors: 16H 32S/T 16383C)
[...]
ada6: 20480MB (41943040 512 byte sectors: 16H 32S/T 16383C)
```
ada0 ist dabei meine Systemplatte, *ada1&2* sind die fiktiven 500GB (in der VM 5GB) Platten, *ada3&4* die fiktiven 1TB Platten und *ada5&6* die 2TB Platten.
Das habe ich ich mir dick und fett in rot auf ein PostIt notiert und diesen an den Rand meines Displays gepappt! Ganz nach dem Motto: Lieber Klebereste entfernen als die falsche Platte plätten! 😉

4.2. FreeNAS 8 erkennt ohne Probleme die FreeNAS 7 *gmirrors*. Also habe ich das alte 2TB-Mirror (hier 20GB da VM) gemountet:
Code:
```
freenas# gmirror list
Geom name: m1
State: COMPLETE
Components: 2
[...]
Providers:
1. Name: mirror/m1
   [...]
Consumers:
1. Name: ada1
   Mediasize: 5368709120 (5.0G)
   [...]
```

```
2. Name: ada2
   Mediasize: 5368709120 (5.0G)
   [...]

Geom name: m2
State: COMPLETE
Components: 2
[...]
Providers:
1. Name: mirror/m2
   Mediasize: 10737417728 (10G)
   [...]
Consumers:
1. Name: ada3
   Mediasize: 10737418240 (10G)
   [...]
2. Name: ada4
   Mediasize: 10737418240 (10G)
   [...]

Geom name: m3
State: COMPLETE
Components: 2
[...]
Providers:
1. Name: mirror/m3
   Mediasize: 21474835968 (20G)
   [...]
Consumers:
1. Name: ada5
   Mediasize: 21474836480 (20G)
   [...]
2. Name: ada6
   Mediasize: 21474836480 (20G)
   [...]
freenas# ls /dev/mirror/
m1      m1p1    m2      m2p1    m3      m3p1
freenas# mkdir /mnt/m3
freenas# mount /dev/mirror/m3p1 /mnt/m3
freenas# ls -lA /mnt/m3
total 4
drwxrwxr-x  2 root  operator  512 May 10 11:35 .snap
drwxr-xr-x  2 root  wheel     512 May 10 11:38 m3-dir
-rw-r--r--  1 root  wheel       0 May 10 11:38 m3-file
```

So, alles da (hier natürlich nur Testdateien und -verzeichnisse).

4.3. Um mein Stripe erstellen zu können, musste ich aus den beiden anderen Mirrors jeweils eine Platte entfernen:

Code:

```
freenas# gmirror remove m1 /dev/ada1
freenas# gmirror remove m2 /dev/ada3
freenas# gmirror status
     Name    Status  Components
mirror/m1  COMPLETE  ada2
mirror/m2  COMPLETE  ada4
mirror/m3  COMPLETE  ada5
                     ada6
freenas# gconcat label -v trans /dev/ada1 /dev/ada3
Metadata value stored on /dev/ada1.
Metadata value stored on /dev/ada3.
```

```
Done.
freenas# newfs /dev/concat/trans
/dev/concat/trans: 15360.0MB (31457276 sectors) block size 16384, fragment
size 2048
        using 84 cylinder groups of 183.72MB, 11758 blks, 23552 inodes.
[...]
freenas# mkdir /mnt/trans
freenas# mount /dev/concat/trans /mnt/trans
freenas# df -h
Filesystem             Size    Used   Avail Capacity  Mounted on
/dev/ufs/FreeNASs1a    458M    399M     22M    95%    /
devfs                  1.0K    1.0K      0B   100%    /dev
/dev/md0               4.4M    2.3M    1.8M    56%    /etc
/dev/md1               686K     10K    622K     2%    /mnt
/dev/md2                75M     10M     58M    15%    /var
/dev/ufs/FreeNASs4      20M    385K     18M     2%    /data
/dev/mirror/m3p1        19G    6.0K     18G     0%    /mnt/m3
/dev/concat/trans       15G    4.0K     13G     0%    /mnt/trans
```

4.4. Nun konnte ich die Daten auf den Stripe übertragen. Ich habe *tar* verwendet, um die Berechtigungen etc. beizubehalten aber ohne Komprimierung:
Code:

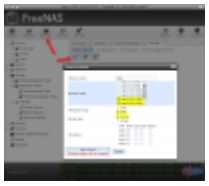
```
freenas# cd /mnt/
freenas# tar cfv trans/m3.tar m3
a m3
a m3/.snap
a m3/m3-dir
a m3/m3-file
```

Auf der echten NAS hat das natürlich ein Weilchen gedauert.

4.5. Die Daten waren auf dem Stripe und ich konnte mein 2TB (respektive 20GB) Mirror zerstören, den Anfang und das Ende der Platten säubern und ein Zpool mit den beiden Platten im WebUI (weil dieses die 4k Sektoren der "echten" WD20EARS automatisch handelt!!!) erstellen:
Code:

```
freenas# umount /mnt/m3
freenas# rmdir /mnt/m3
freenas# gmirror stop m3
freenas# foreach i (ada5 ada6)
foreach? gmirror clear -v /dev/$i
foreach? dd if=/dev/zero of=/dev/$i bs=1m count=1
foreach? end
Metadata cleared on /dev/ada5.
[...]
1048576 bytes transferred in 0.008114 secs (129228881 bytes/sec)
Metadata cleared on /dev/ada6.
[...]
1048576 bytes transferred in 0.008205 secs (127798178 bytes/sec)
```

Und dann im WebUI:



Tipp: In der VM musste ich ich erstmal den Storage Tab schließen und erneut auf den

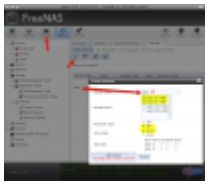
Storage-Button in der Toolbar klicken, was den Tab wieder öffnet, um den erstellten Zpool dann sehen. 
Mit der "echten" NAS hat das problemlos geklappt.

4.7. Das Zpool war jetzt da, also bin ich zurück zum CLI, habe die Daten des Stripes zurück auf den Mirror übertragen
Code:
```
freenas# cd /mnt/data/
freenas# rsync -Pahv /mnt/trans/m3.tar .
```
Ich hätte auch das un-*tar*en können, aber da die Daten später in verschiedenen ZFS-Datasets ihren Platz haben werden, kann ich sie auch nachher explizit in ihr Dataset extrahieren.

4.8. Jetzt waren die Daten vom 500GB Mirror "m1" (in der VM 5GB), welcher nur noch eine Platte hatte, dran. Ich habe wieder tar verwendet, diesmal jedoch mit Komprimierung, da ich an diese Daten wahrscheinlich nicht so schnell ran muss (sie liegen immer noch ge*tar*t da).
Code:
```
freenas# mkdir /mnt/m1
freenas# mount /dev/mirror/m1p1 /mnt/m1
freenas# cd /mnt/
freenas# tar cfvz /mnt/data/m1.tar.gz m1
a m1
a m1/.snap
a m1/m1-dir
a m1/m1-file
```

4.9. Die beiden 500GB-Platten waren jetzt für das Zpool verfügbar. Zunächst musste ich natürlich mein Stripe "trans" und den Mirror "m1" zerstören und die Platten säubern und dann im WebUI ein ZFS-Mirror mit dem **gleichen Namen(!!)**, den ich bei 4.6. verwendet hatte, jedoch mit den beiden 500G Platten, erstellen:
Code:
```
freenas# umount /mnt/m1
freenas# umount /mnt/trans/
freenas# rmdir /mnt/m1/ /mnt/trans/
freenas# gconcat status
        Name  Status  Components
concat/trans      UP  ada1
                      ada3
freenas# gconcat destroy trans
freenas# foreach i ( ada1 ada3 )
foreach? gconcat clear /dev/$i
foreach? dd if=/dev/zero of=/dev/$i bs=1m count=1
foreach? end
[...]
freenas# gmirror status
     Name    Status  Components
mirror/m2  COMPLETE  ada4
mirror/m1  COMPLETE  ada2
freenas# gmirror stop m1
freenas# gmirror clear /dev/ada2
freenas# dd if=/dev/zero of=/dev/ada2 bs=1m count=1
[...]
```

Und dann im WebUI:



In der VM wieder Storage Tab zu und wieder auf und... 

Code:

```
freenas# zpool status
  pool: data
 state: ONLINE
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        data          ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            gpt/ada5  ONLINE       0     0     0
            gpt/ada6  ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            gpt/ada1  ONLINE       0     0     0
            gpt/ada2  ONLINE       0     0     0

errors: No known data errors
```

... Tadaaaa! 😊

4.10. Jetzt war genug Platz auf im Zpool verfügbar, um auch die Daten des 1TB-Mirrors, welcher auch nicht voll war, in den Pool zu übertragen. Also nochmal 4.8., jedoch mit dem 1TB-Mirror und danach wieder das Zpool (=> 4.9.) erweitert, so dass schlussendlich das herauskam:

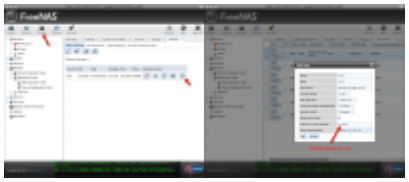
Code:

```
freenas# zpool status
  pool: data
 state: ONLINE
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        data          ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            gpt/ada5  ONLINE       0     0     0
            gpt/ada6  ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            gpt/ada1  ONLINE       0     0     0
            gpt/ada2  ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            gpt/ada3  ONLINE       0     0     0
            gpt/ada4  ONLINE       0     0     0

errors: No known data errors
```

Fertig. Ich hatte, was ich wollte und es ging um einiges schneller, als die Daten von den externen Festplatten wieder aufzuspielen.

4.11. Und was natürlich nicht fehlen darf, ist die Überwachung der Platten. FreeNAS 8 bringt

auch *smartmontools* mit, allerdings sind sie ein wenig versteckt:



Die extra Option *-m root* sorgt dafür, dass *smartd* bei Fehlern den *root* benachrichtig. Wenn die Email-Einstellungen (=>3.2.) richtig gemacht wurden, dann sendet FreeNAS 8 diese an die eingestellte Adresse.

Noch ein paar Worte zu meinem Zpool:
Mein Zpool besteht, wie unter 4.10. zu sehen, aus 3 Mirrors. Im Grunde verwende ich ein RAID 10 (oder RAID 1+0) und trage auch die damit verbundenen Risiken (bis auf die "Write Holes" und der Data Corruption, weil es ja ZFS ist).
Für die, die sich nicht mit RAID Levels auskennen:
Ein RAID 10 ist die Kombination aus RAID 1 und RAID 0. Es ist also ein Verbund (RAID 0) aus mehreren Spiegeln (RAID 1).
Wenn bei einem RAID 0 eines der Komponenten ausfällt, sind die Daten verloren.
Bei einem RAID 10 mit 3 Spiegeln können 3 Festplatten ausfallen, ohne dass es zu Datenverlust kommt. Allerdings müssen es "die richtigen" 3 Festplatten sein, nämlich von jedem Spiegel eine. Fällt ein Spiegel komplett aus, sprich beide Festplatten, sind die Daten futsch.

Genau so verhält es sich auch mit meinem Zpool. Verliere ich ein Mirror komplett, verliere ich den Pool und somit die Daten. Ganz einfach.
Aus diesem Grund wird meine nächste Ausbaustufe (mal gucken, wann ich das in Angriff nehme 😀) ein PCIe SATA Controller und noch 3x Platten, die ich jeweils als eine Art *Hot-Spare für die Mirrors verwenden werde, in dem ich jeweils eine zu den jeweiligen Mirrors hinzufügen werde und so einen sog. 3-way-mirror bekomme. Das einzige, was es dafür zu beachten gilt, ist dass die Platten auf keinen Fall zu klein sein dürfen (ein paar Sektoren reichen - also lieber gleich größere). Dadurch werde ich in jedem Mirror 2 Platten verlieren können, ohne dass der Pool zugrunde geht und bekomme zusätzlich einen Performanceschub (welcher eh Obsolet ist, da mein Flaschenhals das GB-LAN ist 😀). Ob und wie das mit dem FreeNAS 8 WebUI geht - seh'ma dann.*

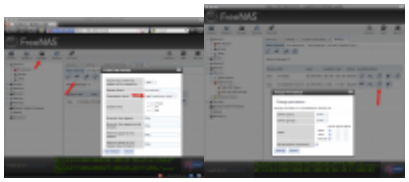
5. Erstellen der ZFS-Datasets

Bei FreeNAS 8 gibt es einen wesentlichen Unterschied zu FreeNAS 7, was die Shares anbelangt: Es können nicht beliebige Pfade, sondern nur über das WebUI gemountete Volumes freigegeben werden. Bei Verwendung von ZFS ist das kein Problem, da ZFS-Datasetzs verwendet werden können.
Was die Datasets anbelangt, gibt es noch einen Unterschied zu FreeNAS 7: FreeNAS 8 unterstützt über das WebUI keine Dataset Children. Ich hoffe, dass das nachgereicht wird.

5.1. Erstellen der Benutzer & Gruppen für die Shares:

Auf die Shares, gerade für Time Machine, sollte mit verschiedenen Benutzern zugegriffen werden. Diese werden folgendermaßen erstellt:
Wird die Einstellung Primary Group auf "---" gelassen, erstellt FreeNAS 8 eine Gruppe mit dem Namen des Benutzers.
Wenn bei "Shell" "nologin" ausgewählt wird, kann sich der Benutzer nicht über SSH anmelden.
Ich füge noch die Benutzer einer gemeinsamen Gruppe hinzu (ich wähle dafür die vordefinierte Gruppe "staff" - man kann auch eine eigene erstellen):

5.2 Datasets für Time Machine:
*Im Grunde könnte, wenn man das möchte, ein einzelnes Dataset für alle TM-Backups verwendet werden. Allerdings ist es dann **nicht** möglich, sich von verschiedenen Macs mit dem selben Benutzer anzumelden und TM-Backups zu erstellen (TM wird beim 2. Mac schon "The disk could not be found" oder so ähnlich melden). Deshalb besser für jeden Mac ein Dataset erstellen. Dann kann auch der gleiche Benutzer verwendet werden.*
Bei den TM-Datasets lohnt es sich, die ZFS maximale gzip-Komprimierung einzusetzen. Vorteil: Die Backups belegen, abhängig davon, wieviele Daten nicht komprimiert werden können (also bereits komprimierte Dateiformate, z.B. mp3's), kann der Speicherbedarf um die Hälfte reduziert werden. Das beste Ergebnis hat einer meiner Testmacs, welcher natürlich keinerlei Multimedia etc. hat: Das gesamte Backupvolumen des Initialbackups beträgt ca. 55G, auf der NAS belegt das TM-Sparsebundle lediglich 23G (~42% der Gesamtgröße!). Mit meinem MBP, der natürlich auch viel Musik, Bilder und weitere komprimierte Dateien hat, schaffe ich lediglich 67% (IMHO immer noch "wow"!).
Damit die Backups nicht unendlich groß werden, setze ich eine Quota fest. Bei meiner FreeNAS habe ich das nach dem Initialbackup gemacht und die Größe des erstellten Sparsebundles im komprimierten Dataset x2 genommen. Beim Testmac, ein Mac mini, habe ich einfach die belegten 23G aufgerundet auf 25G und dann dupliziert, also 50G. Den selben Platz habe ich reserviert, damit die 50 G auf jeden Fall für den Testmac verfügbar sind (in der VM sind's nur 5G).
Die Berechtigungen sollten VOR dem Initialbackup gesetzt werden.



Nachteil: Natürlich geht das nur auf Kosten der Prozessorleistung, RAM und auf die R/W Performance der NAS! Wenn schnell viele Daten in ein Dataset mit maximaler Komprimierung schiebt, dann kommt die NAS richtig ins schwitzen.
Ich verwende redundante TM-Backups, d.h. 1x die Woche schließe ich an jedem Mac eine externe Platte an und mache ein TM-Backup auf ein weiteres Medium. Ich habe deshalb und weil keinerlei akuter Bedarf an Daten von den alten Backups bestand darauf verzichtet, die Backups von den alten FreeNAS 7 Mirrors in die neuen Datasets mit der hohen Komprimierungsrate zu schieben - das hätte länger gedauert, als die TM-Backups erstmal jeweils neu zu erstellen.
Ich werde einfach die Backups auf den Platten nach und nach auf die NAS schieben und dann

bei Gelegenheit wieder umswitchen. Natürlich wird dann, sobald ich die derzeitigen Backups loswerde, ein Stück fehlen (ca. 7 Tage) aber das ist in Ordnung.
Die Initialbackups über das Netzwerk hat die NAS übrigens concurrent und ohne Zicken geschluckt. Mit der Datenmenge, die per GB-LAN reinkommt, wird meine NAS kaum überfordert werden. Wie gesagt: das ist der Flaschenhals.

5.3. Weitere Datasets werden genau so erstellt, wie unter 5.2 erklärt. Hierbei sollte bedacht werden, was für Daten in den Datasets gespeichert werden wird und die Komprimierung entsprechend einstellen. Habe ich bspw. ein Dataset für eine iTunes Library, die nur aus mp3's und m4a's besteht, brauche ich dafür eigentlich keine Komprimierung. Habe ich 90% wav's in meiner Library, könnte es wieder Sinn machen... aber an die benötigten Ressourcen für die Komprimierung denken!
Desweiteren sollte folgendes bedacht werden, dass Datasets sind wie eigene Partitionen zu betrachten sind.
-Es können keine Hardlinks zw. Datasets erstellt werden. Das wäre ein Cross-device Link.
-Das verschieben von Dateien zw. Datasets ist wie das verschieben von Daten zw. verschiedenen Partitionen: es wird nicht nur der Inode-Eintrag geändert, sondern die komplette Datei wird auf das andere Dataset kopiert und dann im ursprünglichen gelöscht. D.h. wenn Daten auf der NAS hin und her geschoben werden sollen, empfiehlt es sich, diese in einem Dataset zu speichern und nicht in verschiedene.

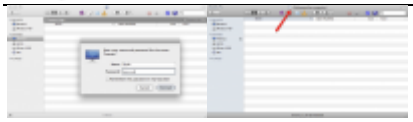
6. Erstellen der Shares

6.1. Time Machine Shares erstellen:
Am einfachsten mit durch diese beiden Screenshots erklärt:

6.2. Weitere AFP-Freigaben erstellen funktioniert genau so wie bei den TM-Shares, nur dass man die Option Disk Descovery einfach aus lässt. Ich verwende gerne für die "normalen" AFP-Shares zusätzlich die No. AppleDouble-Option.

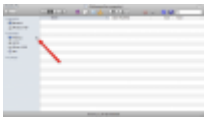
6.3. AFP einschalten:

6.4. Time Machine Shares nutzen:
Im Gegensatz zu FreeNAS 7 scheint FreeNAS 8 die TM-Shares nicht so... ähm... "anzukündigen", dass Time Machine sie direkt aus dem Netzwerk sehen kann. Es muss erstmal die TM-Freigabe gemountet werden und dann kann sie in TM genutzt werden:
6.4.1. Das entsprechende TM-Share der Freenas mit dem Finder mounten

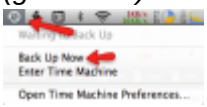


6.4.2. Die TM-Einstellungen öffnen, "Select Disk" (bzw. was auch immer auf Deutsch da steht) klicken, das TM-Volume der NAS auswählen, Schalter auf "An" stellen.

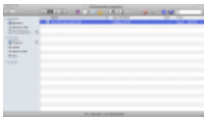
6.4.3. Im Finder das TM-Share wieder unmounten



6.4.4. Um den Start des Backups zu beschleunigen, auf das TM-Icon in der Menüleiste klicken (lässt sich in den TM-Einstellungen an- und ausschalten) und auf "Backup jetzt erstellen" (glaub' ich) klicken. TM mountet selbstständig die TM-Freigabe und legt los.



6.4.5. Beobachten und warten 
6.4.6. Wenn alles gut gegangen ist, dann hat TM erfolgreich ein Backup erstellt und auf dem TM-Share befindet sich jetzt ein Sparsebundle:



6.4.7. Zum Testen nochmal, falls gemountet, das TM-Share unmounten und in die Time Machine gehen (/Programme/Time Machine.app oder über das Menüicon).

6.5. Zuverlässigkeit der TM-Backups
Es wird oft gemunkelt, TM-Backups auf nicht-□-Lösungen wären nicht zuverlässig. Ich kann das nicht bestätigen. Zum einen habe ich schon öfters Macs von TM-Backups auf einer FreeNAS 7 wiederhergestellt, zum anderen habe ich auch schon TM-Backups auf Time Capsules erlebt, die nicht funktionierten.
Meiner Erfahrung nach funktionieren TM-Backups, egal ob von einer TC, einer FreeNAS oder einem anderen Gerät zur Verfügung gestellt, solange TM zuverlässig auf das TM-Share sichert und die Datenintegrität gegeben ist (bei FreeNAS 8 mit ZFS sorgt eben ZFS für die Datenintegrität des Sparsebundles). Ich lasse mich gerne eines besseren belehren, falls jemand anderweitige Erfahrungen gemacht hat.
Weil es trotzdem ein heikles Thema ist, habe ich die Erstellung sowie Wiederherstellung von einem TM-Backup auf und von der FreeNAS 8 auf 10.5.8 und 10.6.7 getestet: Funktioniert!
Ich lasse es mir dennoch nicht nehmen, auch weiterhin 1x pro Woche ein TM-Backup auf eine externe Festplatte zu machen. Eine weitere Redundanzstufe schadet keinem Backup.

6.6. SBM und NFS-Shares werden ähnlich erstellt wie die AFP-Shares. Auch hierfür muss der entsprechende Dienst (Service) gestartet werden und die Shares konfiguriert. Sollte eigentlich selbsterklärend sein.

7. FreeNAS 8 erweitern

Vorweg: FreeNAS 8 muss nicht erweitert werden. Diejenigen, die mit den über das WebUI verfügbaren Features zufrieden sind und eigentlich nichts mehr vermissen, brauchen das nicht. Unbedarfte sollten vielleicht auch die Finger davon lassen, da dadurch Probleme entstehen könnten, die auch erstmal gefunden werden wollen.
Wenn hier nicht weitergemacht wird, bitte daran denken den direkten root-Login über SSH (siehe 3.6) oder SSH komplett zu deaktivieren.

Zum erweitern von FreeNAS 8 wird folgendes benötig:
-UnionFS (ist "Onboard")
-ein rc.d-Script
-ein UFS-Volume, da UnionFS nicht von Zpools funktioniert
-Eine funktionierende Internetverbindung und Netzwerkkonfiguration der FreeNAS.

Die ursprüngliche Anleitung für FreeNAS 7 findet ist hier zu finden.

Und so geht's für FreeNAS 8:
7.1. Erstellen eines 5GB zvols mit UFS+SoftUpdates und mounten:
Code:

```
freenas# zfs create -V 5G data/opt
freenas# newfs -U /dev/zvol/data/opt
/dev/zvol/data/opt: 5120.0MB (10485760 sectors) block size 16384, fragment
size 2048
        using 28 cylinder groups of 183.72MB, 11758 blks, 23552 inodes.
        with soft updates
[...]
freenas# mkdir /mnt/opt
freenas# mount /dev/zvol/data/opt /mnt/opt/
freenas# mount -t ufs
[...]
/dev/zvol/data/opt on /mnt/opt (ufs, local, soft-updates)
```

Wunderbar.

7.2. Jetzt erstellt man auf /mnt/opt die benötigten Verzeichnisse, die später mit UnionFS auf / gemountet werden sollen:
Code:

```
freenas# cd /mnt/opt/
freenas# mkdir -vp usr etc home/fyysh/.ssh root/.ssh var_db var_cron
usr
etc
home
home/fyysh
home/fyysh/.ssh
root
root/.ssh
var_db
var_cron
```

Vorsicht: /tmp und /var sollten nicht UnionFS gemountet werden bzw. es benötigt nähere Betrachtung. Ich hatte diese beiden anfänglich ebenfalls UnionFS-Gemountet, auch nur /var (/tmp ist ein Link auf /var/tmp) und das hatte zur Folge, dass die FreeNAS nach einiger Zeit unresponsive wurde. Ich bin dazu übergegangen, nur das aus /var zu mounten, was ich tatsächlich benötige, nämlich /var/db und /var/cron. Seitdem funktioniert's problemlos.

7.3. Anschließend erstellt man noch die SSH-Keys, bash- und cshrc's etc., denkt an die Berechtigungen usw.
So sieht das dann bei mir aus:
Code:

```
freenas#  ls -RlA /mnt/opt/home/fyysh/
-rwxr-xr-x  1 fyysh  fyysh    26 Apr  9 15:09 .bash_aliases
-rwxr-xr-x  1 fyysh  fyysh  1168 Apr 24 01:10 .bashrc
-rwxr-xr-x  1 fyysh  fyysh  2764 May  8 12:12 .cshrc
-rw-r--r--  1 fyysh  fyysh  1918 Apr 11 10:01 .inputrc
drwx------  2 fyysh  fyysh   512 Feb 16 03:00 .ssh

home/fyysh/.ssh:
total 22
-rw-------  1 fyysh  fyysh  4515 Oct  8  2010 authorized_keys
-rw-r--r--  1 fyysh  fyysh   495 Dec 11 00:29 config
-rw-------  1 fyysh  fyysh  1675 Oct  8  2010 id_rsa
-rw-------  1 fyysh  fyysh   411 Oct  8  2010 id_rsa.pub
```

7.4. Wenn ich fertig bin mit meinen Anpassungen, umounte ich /mnt/opt wieder und lösche das mnt-Verzeichnis:
Code:

```
freenas# umount /mnt/opt/
freenas# rmdir /mnt/opt/
```

7.5. Es ist Zeit für das rc.d-Script.
Es lädt einfach alle Verzeichnisse unter /mnt/opt und mountet sie nach /
Die Variablen UNION_ZVOL_DEV und UNION_ZVOL_MP müssen ggf. angepasst werden.
Mein Script führt fsck_ufs aus, obwohl ich Softupdates aktiviert habe. Dauert bei 5G nicht wirklich lange und schaden tut's auch nicht.
Übrigens: Falls jemand ein besseres Script hat: bitte zur Verfügung stellen. 😊
Das Script:
Code:

```
#!/bin/sh
#
# $FreeBSD$
#

# REQUIRE: ix-httpd LOGIN
# KEYWORD: nojail shutdown


. /etc/rc.subr

name="unionfs-magic"
start_cmd="${name}_start"
stop_cmd="unionfs-magic_stop"

UNION_ZVOL_DEV="/dev/zvol/data/opt"
UNION_ZVOL_MP="/mnt/opt"
UNION_OPTS="copymode=transparent"     # mount_unionfs options, comma
separated


unionfs-magic_start()
{
```

```
        # check if volume is already mounted, if yes, umount it
        echo -n "Checking if ${UNION_ZVOL_MP} is already mounted... "
        if mount | grep -q "${UNION_ZVOL_DEV} on ${UNION_ZVOL_MP}"; then
                echo "YES"
                echo -n "Unmounting ${UNION_ZVOL_MP}... "
                if umount -f "${UNION_ZVOL_MP}"; then
                        echo "SUCCESS"
                else
                        echo "FAIL"
                        echo "An error ocured, cannot continue script."
                        return 1
                fi
        fi

        # fsck the volume
        echo "Performing fsck_ufs -p on ${UNION_ZVOL_DEV}"
        fsck_ufs -y ${UNION_ZVOL_DEV} || \
         { echo "An error ocured. Cannot continue script"; return 1; }
        echo ""

        # check if the mountpoint exists or create it
        echo -n "Checking if mountpoint ${UNION_ZVOL_MP} exists... "
        if test -d "${UNION_ZVOL_MP}"; then
                echo "OK"
        else
                 if mkdir "${UNION_ZVOL_MP}"; then
                        echo "CREATED"
                else
                        echo "Cannot create mountpoint ${UNION_ZVOL_MP}"
                        echo "An error ocured, cannot continue script."
                        return 1
                fi
        fi


        # (re)mount the volume to the mountpoint
        echo -n "Mounting ${UNION_ZVOL_DEV} to ${UNION_ZVOL_MP}... "
        if mount "${UNION_ZVOL_DEV}" "${UNION_ZVOL_MP}"; then
                echo "SUCCESS"
        else
                echo "FAIL"
                echo "An error ocured, cannot continue script."
                return 1
        fi


        # mount all directories found in the mountpoint using unionsfs
        echo "Using unionfs to mount directories from ${UNION_ZVOL_MP}"
        local IFS='
'
        for i in $(ls -d1 "${UNION_ZVOL_MP}"/*);do
                # convert _ to /
                UNION_DIR="/$(basename "$i"|tr _ /)"

                # check if mountpoint exists or create it
                echo -n "Checking if mountpoint $UNION_DIR exists... "
                if [ ! -d "$UNION_DIR" ]; then
                        mount -uw /
                        mkdir -p "$UNION_DIR"
                        mount -ur /
                        echo "CREATED"
```

```
                else
                        echo "OK"
                fi

                # mount with unionfs
                echo -n "Mounting $i to $UNION_DIR... "
                mount -t unionfs -o $UNION_OPTS "$i" "$UNION_DIR" && echo
"SUCCESS" || echo "FAIL"
        done
}


unionfs-magic_stop()
{
        echo "Unmounting unionfs mounts..."
        for i in $(mount -t unionfs | awk '{print $3}');do umount -fv $i;done
        echo ""
        echo "Unmounting ${UNION_ZVOL_MP}"
        umount -fv ${UNION_ZVOL_MP} && rm -rfv ${UNION_ZVOL_MP}
}


load_rc_config $name
run_rc_command "$1"
```

Das Skript erstelle ich lokal und übertrage es per scp auf die NAS

Code:

```
$> scp unionfs-magic root@192.168.254.223:/mnt/data
root@192.168.254.223's password:
unionfs-magic                                                 100% 2343
2.3KB/s   00:00
```

Anschließend muss ich auf der NAS / rw mounten, das script in /conf/base/etc/rc.d/ ablegen, es ausführbar machen und / wieder ro mounten

Code:

```
freenas# mount -uw /
freenas# cp /mnt/data/unionfs-magic /conf/base/etc/rc.d/
freenas# chmod +x /conf/base/etc/rc.d/unionfs-magic
freenas# mount -ur /
```

Nun testen (diesmal direkt von /conf/base/etc/rc.d/unionfs-magic, nach einem Neustart kann es aus etc/rc.d/unionfs-magic ausgeführt werden):

Code:

```
freenas# /conf/base/etc/rc.d/unionfs-magic start
Checking if /mnt/opt is already mounted... Performing fsck_ufs -p on
/dev/zvol/data/opt
** /dev/zvol/data/opt
** Last Mounted on /mnt/opt
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
9 files, 9 used, 2538510 free (46 frags, 317308 blocks, 0.0% fragmentation)

***** FILE SYSTEM IS CLEAN *****

Checking if mountpoint /mnt/opt exists... CREATED
Mounting /dev/zvol/data/opt to /mnt/opt... SUCCESS
Using unionfs to mount directories from /mnt/opt
Checking if mountpoint /etc exists... OK
Mounting /mnt/opt/etc to /etc... SUCCESS
```

```
Checking if mountpoint /home exists... CREATED
Mounting /mnt/opt/home to /home... SUCCESS
Checking if mountpoint /root exists... OK
Mounting /mnt/opt/root to /root... SUCCESS
Checking if mountpoint /usr exists... OK
Mounting /mnt/opt/usr to /usr... SUCCESS
Checking if mountpoint /var/cron exists... OK
Mounting /mnt/opt/var_cron to /var/cron... SUCCESS
Checking if mountpoint /var/db exists... OK
Mounting /mnt/opt/var_db to /var/db... SUCCESS
freenas# mount
[...]
data on /mnt/data (zfs, local)
/dev/zvol/data/opt on /mnt/opt (ufs, local, soft-updates)
<above>:/mnt/opt/etc on /etc (unionfs, local)
<above>:/mnt/opt/home on /home (unionfs, local)
<above>:/mnt/opt/root on /root (unionfs, local)
<above>:/mnt/opt/usr on /usr (unionfs, local)
<above>:/mnt/opt/var_cron on /var/cron (unionfs, local)
<above>:/mnt/opt/var_db on /var/db (unionfs, local)
```

Passt. Um alles wieder sauber zu unmounten, müsste ich übrigens /conf/base/etc/rc.d/unionfs-magic stop respektive nach Neutstart etc/rc.d/unionfs-magic stop ausführen.
In /var/log/console.log steht der Output des Scripts während des Starts.
Noch ein paar Hinweise zum Script:
-Wird in der Variable UNION_OPTS noch die Option below eingetragen, werden die "echten" Freenas-Dateien "oben auf" gemountet (siehe dazu die Manpage von mount_unionfs)
-Wird bei den KEYWORDS noch zusätzlich nostart verwendet, wird das script nicht beim Start geladen.
-Um in ein Subverzeichnise zu unionfsen, muss das Verzeichnis in /mnt/opt mit einem _ erstellt werden. Wie man im Output des Script sieht, wird /mnt/opt/var_db nach /var/db gemountet.
-Sollte die Installation eines Ports die Freenas zerschossen haben, kann mit der Installations-CD oder mit einem anderen USB-Stick mit FreeNAS 8 oder auch FreeBSD die Maschine gestartet werden, anschließend die Datenpartition von der FreeNAS mounten (bei mir ist das /dev/ad0s1a in der VM bzw. /dev/ad6s1a in der echten FreeNAS) und das Script um das KEYWORD "nostart" erweitert werden oder gar gelöscht werden. FreeNAS sollte dann wieder ganz normal starten, da das eigentliche System ansonsten nicht angefasst wurde.

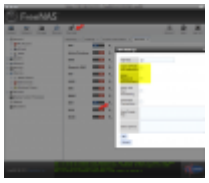
7.6. Jetzt das Homeverzeichnis für den User, der über SSH verwaltet, bei mir "fyysh", im WebUI ändern um und dann neu starten:

7.7. Testen, ob Key-auth funktionert (wenn es funktioniert, funktionieren die UnionFS mounts)
Code:

```
$> ssh 192.168.254.223
[...]
Welcome to my fancy FreeNAS 8
%mount
[...]
data on /mnt/data (zfs, local)
```

```
data/tm-macmini on /mnt/data/tm-macmini (zfs, local)
/dev/zvol/data/opt on /mnt/opt (ufs, local, soft-updates)
<above>:/mnt/opt/etc on /etc (unionfs, local)
<above>:/mnt/opt/home on /home (unionfs, local)
<above>:/mnt/opt/root on /root (unionfs, local)
<above>:/mnt/opt/usr on /usr (unionfs, local)
<above>:/mnt/opt/var_cron on /var/cron (unionfs, local)
<above>:/mnt/opt/var_db on /var/db (unionfs, local)
```

Super. Jetzt kann ich die Passwortanmeldung per SSH deaktivieren:



Und da ich mit meiner Konfiguration der FreeNAS über das WebUI hier fertig war, habe ich auch gleich selbige gesichert.

7.8. Wieder mit SSH auf die NAS und einige Extras installieren/Anpassungen machen.
Kleine Warnung: NICHT den Midnight Commander-Port mc installieren! Das zerschießt das WebUI. mc-light funktioniert problemlos.
Code:

```
$> ssh 192.168.254.223
[...]
Welcome to my fancy FreeNAS 8
%su
Password:
freenas# pkg_add -r -v zip unzip unrar sudo wget curl subversion p7zip most
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-8.2-
release/Latest/zip.tbz... Done.
[...]
```

Das installieren der infos schlägt natürlich fehl, weil die Dokumentation nicht installiert ist. Das würde die FreeNAS-Images unnötig aufblähen.

7.9. Eine Kleine kosmetische Änderung:
Damit die FreeNAS in der Seitenleiste des Finders ein bisschen pfiffiger ausschaut, muss man noch eine Datei editieren (Quelle). Da /usr schon geunionfst, ist, kann man das direkt ohne den / rw zu mounten machen:
Code:

```
freenas# nano /usr/local/etc/avahi/services/afp.service
```

Und dann muss das so aussehen:
Code:

```
[...]
<service-group>

        <name replace-wildcards="yes">%h</name>

        <service>
                <type>_afpovertcp._tcp</type>
                <port>548</port>
        </service>
        <service>
                <type>_device-info._tcp</type>
                <port>0</port>
                <txt-record>model=RackMac</txt-record>
        </service>
```

```
</service-group>
```

Anschließend lädt man den avahi-daemon neu
Code:

```
freenas# avahi-daemon -r
```

Und um den Effekt direkt im Finder zu sehen, schaltet man im WebUI unter "Sevices" AFP aus und gleich wieder ein. Die FreeNAS verschwindet kurz aus dem Finder und taucht gleich wieder auf.
Vorher und nachher:

7.10. Noch 3 Anpassungen, die mir pers. das Leben erleichten.
*Aber Vorsicht: die ersten beiden (Anpassung von /usr/local/etc/sudoers und /etc/pam.d/su) können ganz leicht dicke Sicherheitslücken werden. Bitte nur machen, wenn du weißt, was du tust! Ich mache das ausschließlich auf meiner **privaten** NAS zuhause und habe auch das Konsolenmenü ausgeschaltet, d.h. es gibt keinen Zugriff zur Shell ohne Passwort und SSH ohne Key ist nicht!*
Durch folgende Modifikation in /usr/local/etc/sudoers (sudo ist nicht an Board und muss nachinstalliert werden) kann der Benutzer "fyysh" ohne Abfrage des Passworts sudo'en:
Code:

```
[...]
##
## User privilege specification
##
root    ALL=(ALL) ALL
fyysh   ALL=(ALL) NOPASSWD: ALL
[...]
```

Die Änderung ist in Zeile 80f vorzunehmen.
Folgende Modifikation ermöglicht ein su ohne Abfrage des Passworts:
Code:

```
[...]
# auth
auth            sufficient      pam_rootok.so           no_warn
auth            sufficient      pam_self.so             no_warn
auth            requisite       pam_group.so            no_warn group=wheel
root_only fail_safe
# auth          include         system[...]
```

Einfach Zeile 11 auskommentieren.
Nochmal: Das nur machen, wenn der Rest abgesichert ist!
Die dritte Anpassung ist das Linken der ganzen persönlichen rc- & etc. Dateien, also bashrc, cshrc etc., aus dem Homeverzeichnis des Verwaltungsbenutzers, bei mir immer noch "fyysh", nach /root, so dass die Aliase, Funktionen und andere Anpassungen auch nach einem su funktionieren. Nur .ssh darf nicht gelinkt werden.

7.11. Und zum Schluss eine Warnung:
Vor dem durchführen von Einstellungen im WebUI oder einem einem Firmaware-Update sollte man besser die UnionFS-Mounts unmounten, also ein etc/rc.d/unionfs-magic stop durchführen.

Bei den Updates muss man sowieso aufpassen. Die UnionFS-Mounts werden "above" gemountet, d.h. zuerst wird im /mnt/opt geguckt und dann erst im "wahren" Verzeichnis. Das kann zur Folge haben, dass nach einem Update eine alte Konfiguration für irgendetwas verwendet wird, wo doch eigentlich eine neue nötig wäre. Der Transparent-Mode von UnionFS sorgt zwar dafür, dass nur Dateien in /mnt/opt kommen, die nachinstalliert wurden, es können aber auch durch Änderungen über die WebUI Änderungen an solchen Dateien stattfinden. Man sollte nach einem Update auf jeden Fall ein Auge darauf haben.

Man könnte natürlich auch direkt in die FreeNAS installieren, aber 1. ist der Platz begrenzt und 2. hat man dann keine Möglichkeit, das alles einfach durch das deaktivieren des Scripts rückgängig zu machen, falls doch etwas zerschossen wurde. Außerdem müsste man die Anpassungen nach einem Update komplett neu durchführen.

8. Was ist noch möglich?

Natürlich mache ich die unter 7 aufgeführten Anpassungen nicht zum Spaß. Meine FreeNAS führt, Cron-getriggert, einiges aus, z.B.:
-Backups meiner Online-Server über rsync mit tw. Versionierung mittels ZFS Snapshots (vor FreeNAS 8 hatte ich das mit rsnapsot gemacht - ZFS ist deutlich bequemer)
-Spiegelt meine wichtigsten Daten auf den Online Backupserver
-etc.

FreeNAS 7 habe ich noch zusätzlich als internen Web- und mySQL-Server verwendet. Das bin ich mit FreeNAS 8 noch nicht angegangen, müsste aber auch gehen. Und auch Unison habe ich verwendet.
Klar, könnte das auch nachinstalliert werden, ich warte jedoch lieber auf FreeNAS 8.1, da die Roadmap doch interessant aussieht und so eine funktionierende, schöne WebUI halt doch sehr bequem ist. Deswegen verzichte ich auch auf das Eingehen in Cronjobs etc.. Schließlich bekommen diejenigen, die sie einsetzen würden, auch so heraus, wie es geht. Die Voraussetzungen dafür, dass sie einen reboot überleben, sind mit den UnionFS-Mounts gemacht.

9. Mein FreeNAS 8.0 Fazit

Ich fasse mich zur Abwechslung kurz:
Bisher ist FreeNAS 8 auf die wesentlichsten bekannten Features aus FreeNAS 7 beschränkt und kommt mit einem neuen, zwar zugegebenermaßen gewöhnungsbedürftigen aber doch guten und "fancy" WebUI daher. Zum Teil ist die Bedienung des WebUI nicht immer ganz intuitiv und so manch mögliche Fehlerursache ist noch nicht abgefangen (ich bin über einige gestolpert), was im Fehlschlagen einer Aktion ohne zunächst erkennbaren Grund enden kann - man ist gezwungen einmal genauer hinzuschauen. Das wird sich aber mit der Zeit geben und auch vieles von dem hier erwähnten wird wahrscheinlich obsolet werden mit der Zeit.

Über die Zuverlässigkeit von FreeNAS 8 kann ich noch nicht viel sagen. Bevor das alte Mainboard abgeraucht ist, hatte meine alte FreeNAS 7 eine Uptime von ca. 150 Tagen und ich

schätze vor diesem Neustart ist sie noch länger gelaufen. Davon ist FreeNAS 8 mit seinen 5 Tagen noch weit entfernt.

Für mich ist FreeNAS 8 zweifelsohne das nächste Level von FreeNAS. Gerade wenn man die Roadmap betrachtet, muss man doch eigentlich genüsslich mit der Zunge schnalzen. Die Devils von iXsystems, Josh Paetzel und sein Team, werden das schon schaukeln und uns mit den kommenden Updates eine unglaubliche NAS liefern - und das für Umsonst (bis auf die HW natürlich). Danke dafür!

Für die kommerziellen Lösungen wird es, Preis-/Leistungstechnisch schwierig mitzuhalten, denke ich - die Stromrechnung mal außer acht lassend. 😉

Quelle: https://www.apfeltalk.de/community/threads/freenas-8-0-ein-how-to.363353/