ZFS

Das Zetabyte Filesystem. Der Heilsbringer für alle kleineren und mittleren Installationen. Trojanersicher und extrem vielseitig!

- **Schutz mit ZFS**
- **ZFS Befehle**
- **ZFS Begriffe**
- **Vergleich von True NAS und Proxmox VE als Storage Server**
- **ZFS RAID**
- **ZFS Optimierung und Todsünden**
- **ZFS Import - Importieren von ZPOOLS**
- **ZFS Arc oder Arcsize**
- **Neue Anzeigen der Eigenschaften der ZSF-Datasets**
- **SOURCE-Werte**
- **HD-Identifikation unter ZFS**
- **Aktualisieren von ZFS-Speicher-Pools**
- **Dateiwiederherstellung von ZVOLs in Proxmox**
- **Sharing unter ZFS - SAN Protkolle**
- **Proxmox VM mit ZFS Auto-Snapshots klonen**
- **Schutz mit ZFS für VSphere und Hyper-V**
- **ZFS Ashift**

Schutz mit ZFS



Wie eine Variable mit drei Buchstaben alles ändern kann!

Zetabyte Filesystem statt proprietärer Raid Controller oder Blackboxen ohne Kontrolle

ZFS ist ein von Sun Microsystems entwickeltes transaktionales Dateisystem, das zahlreiche Erweiterungen für die Verwendung im Server- und Rechenzentrumsbereich enthält. Hierzu zählen die vergleichsweise große maximale Dateisystemgröße, eine einfache Verwaltung selbst komplexer Konfigurationen, die integrierten RAID-Funktionalitäten, das Volume-Management sowie der prüfsummenbasierte Schutz vor Datenübertragungsfehlern. Einfach formuliert: Der Verbund der Festplatten wird von der gleichen Software geregelt die auch die Inhalte der Festplatten bereitstellt. Dadurch ist der Austausch von Datenträgern vor dem Ausfall bereits ohne Verlust der Redundanz möglich. Die vektorbasierte Snapshottechnoligie ermöglicht tausende Versionen ohne Last bereitzustellen. Im Notfall können alte Systemstände gestartet, extrahiert oder analysiert werden. Ohne Bereitstellungszeit! Die Veränderungen der Snapshots können einfach per Push oder trojanersicherem Pull auf beliebig viele Systeme übertragen werden. Im Vergleich zu einem Backup ist ein Replikat kein Versprechen auf eine Wiederherstellung, es ist eine Garantie! Dank ZFS endlich nie wieder Festplatten durch die Gegend tragen und Langzeitarchive aufbauen!

Unsere wichtigsten Lösungen hierzu sind Proxmox VE und TrueNAS Scale/Core, welche beide ZFS integriert haben und daher über allen traditionellen, oft überladenen, Lösungen stehen. Beide Systeme sind Open Source und können kostenfrei auf nahezu jeder Hardware genutzt werden.



Egal welche ergänzende Backupsoftware Sie nutzen...

mit ZFS als Basis können sie Ihre Backups vor Kompromittierung schützen. Damit es erst gar nicht so weit kommen kann, nutzen wir eine Technik die wir liebevoll Miyagimethode nennen. Die beste Verteidigung gegen Angriffe ist einfach nicht da zu sein. Unsere Ultimate Backupserver sind nicht nur unerreichbar für Angreifer, sondern die meiste Zeit ausgeschaltet und damit unangreifbar! Final schliessen wir das Konzept mit einer Überwachungsmethode die das Vergessen von Sicherungen unmöglich macht und durch Prüfsummen unsere Replikate ohne Fehlerquote eindeutig bestätigt. Wir haben also kein Versprechen auf Sicherung, sondern eine Garantie!

Schon bei der ersten Störung bezahlt sich das komplette System

Eine Datensicherung ist eine Methode die im Notfall die Zeit und auch den Platz für eine Wiederherstellung benötigt. Im Schnitt dauert eine Rücksicherung, wenn es bei einer bleibt, mehrere Stunden bis mehrere Tage. Gerechnet auf den Ausfall der Mitarbeiter, der verlorenen Aufträge und dem immensen Schaden an der eigenen Reputation, ist die Investition in ein kostengünstiges ZFS-System schon fast lächerlich im Verhältnis zu dem Stress eines längeren Ausfalls. Es gibt keine Wiederherstellungszeit oder extra Platzbedarf!

ZFS Befehle

ZFS Befehle

Beispiel für eine selbst erstellte Eigenschaft:

zfs set zfs:backup=basename tank/data/vm-1066-disk-0

setzt die Eigenschaft zfs:backup Option basename auf local für die angegebene Disk0

Befehl.....	Erklärung
zpool status	Zeigt den Zustand der ZPOOLS
zpool iostat 1	Zeigt Perfomance Lesen/Schreiben, 1=fortwährend
zfs list	Zeigt die eingerichteten Datasets
zfs list -o space	[-o name|-o compression|...] Gute Übersicht der Datasets inkl. Speicherverbrauch Snapshots (Option `-t snapshot` zeigt nur Snapshots, `-H` ohne Überschrift für Scripts)
zfs get written	Zeigt alle geschriebenen *datasets* inkl. Snapshots
zfs get written | grep 402-disk	Erweiterung mit `grep` um einzelne VMs zu finden (hier 401)
zfs get all rpool/data/<ort>	*all* zeigt alle Werte der Dateneigenschaften, z.B. Zugriffsrechte oder Snapshots
zfs get compression rpool/data/<ort>	[creation|compression|...] Zeigt Eigenschaft (hier: Kompression) eines *datasets*
zfs get -r <eigenschaft> rpool/data/<ort>	Zeigt Eigenschaft *rekursiv* inkl. Unter-Datasets
zfs set <eigenschaft>=[on|off] rpool/data/<ort>	Ein- und Ausschalten einer Eigenschaft **Achtung bei -r!** (siehe **ZFS Options**)
zfs allow	Um Eigenschaften des Hauptverzeichnisses zu delegieren
zfs inherit -r <eigenschaft> rpool/data/<ort>	Der Unterbefehl *inherit* wird *rekursiv* angewendet, wenn die Option *-r* angegeben wird (Achtung: Überschreibt Eigenschaften)
zfs create rpool/data/<ort>	Erstellt ZFS Pool z.B. *rpool/data/repl* für Replikationsverzeichnis
zfs destroy pool/data/<ort oder VMID>	Löscht den ZFS-Pool (bei vorhanden Snapshots nicht möglich)
zfs destroy -r rpool/data/<ort oder VMID>	Löscht den ZFS-Pool inkl. aller Snapshots **unwiderruflich**

ZFS Begriffe

Begriff.....	Erklärung
ZPOOL	Allgemeiner Name für einen Speicher-*Pool* von HDs (von einer einzigen Festplatte [einzelne *vdev*]) bis raid-z3 [mehrere *vdes*], einem Software RAID-System auf ZFS-Basis
rpool	Standardmäßig angelegter *ROOT-ZPOOL* bei Proxmox-Installationen
zfs Dataset	Komplett individuelles Dateisystem z.B. unterhalb vom *ROOT ZFS Datasets* oder weiterer ZPOOLs (z.B. *mypool*). Dieses entsteht beim `zfs create` des Raids, beispiel **rpool/data** - Dort legt man üblicherweise Dateien ab, diese können auch *VMDK*, *RAW*oder *QCOW* für VMs enthalten
ZVOL	Reservierung von Blöcken, erreichbar über den */dev/*-Ordner (Beispiel: `/dev/zvol/rpool`), und wird unter /dev/zvol/rpool/data nochmals detailiert symbolisch verlinkt. Üblicher Zugriff erfolgt über den PVE Manager der diese Volumes den VMs zuweist und dort auch verwaltet

Vergleich von True NAS und Proxmox VE als Storage Server

1. TrueNAS (ehemals FreeNAS)

1. Reiner Storage Server für NAS und SAN Aufgaben
2. Rudimentäres Containering und unbrauchbare Virtualisierung
3. Solider Unterbau basiert auf FreeBSD, gilt als noch stabiler als Linux
4. Reine Appliance, Bedienung erfolgt ausschliesslich über Webinterface
5. **Kernfunktionen**
   - Samba NAS inkl. AD Integration und SID
   - NFS NAS/SAN inkl. Zugriffskontrolle
   - iSCSI SAN inkl Zugriffskontrolle
6. **Indikation für Einsatz**
   - Ergänzung als ZFS Fileserver mit oder ohne AD Integration
   - SAN Server mit NFS für höhere Anzahl von PVE Systemen mit HA Bedarf
   - SAN Server mit iSCSI für höchsten HA Bedarf
   - SAN Server mit iSCSI für einfache Backup oder OS Aufgaben
   - Vollständiges Monitoring und Webgui für Snapshots und Replikation, sowie den Systemzustand

2. Proxmox VE

1. Gemischtes System für Virtualisierung und ZFS Storage
2. Umfangreiche Virtualisierung und Containering-Lösung
3. Moderner Unterbau mit aktuellem Debian Linux
4. Reichhaltige Funktionen in allen Bereichen kommen schnell an ihre Grenzen und werden durch eigene Kommandozeilenlösungen erweitert
5. **Kernfunktionen**
   - KVM Virtualisierung und LXC Containering
   - ZFS Storagesystem für Software Raid und rudimentärer Snapshot- und Replikationslösung
   - Optional: NFS-Kernel Server
   - Optional: iSCSI Target
   - Optional: erweitertes ZFS-Management mit **Sysops** Hauslösungen

6. **Indikation für Einsatz**
    - Kleinere Anzahl von PVE Systemen mit potenter Aufnahme von Datenträgern
    - Hoher Anspruch an Snapshotvorhaltung
    - Leichte Replikation der Daten auf zweites System und eventuell weitere Ziele
    - Applikationsbasierte Datensicherung
    - Eigenes **Monitoring** und Reporting bevorzugt
    - Einfacheres Deployment und Wiederherstellung

ZFS RAID

Übersicht ZFS-RAID

- RAID-1 (mirror) *Empfohlen z.B. für reines Proxmox-System ohne VMs*
- RAID-Z1 (1 Parity-Bit, ~RAID 5)
- RAID-Z2 (2 Parity-Bits, ~RAID 6)
- RAID-Z3 (3 Parity-Bits)

RaidZ(2,3)

- Sinnvoll für Datenablage und gelegentlichen Zugriff
- Oft in Kombination mit harter Kompression im NAS Betrieb
- Bei Erweiterung müssen alle Datenträger im Verbund getauscht werden
- Bei PVE als Datastore dringend 16k Blocksize für neue Systeme hinterlegen, da sonst extremer Overhead (siehe Warnung)
- x-y Verlust an Platz für Redundanz

Raid Z+1

- Eventuelle Kette aus mehreren RaidZ denkbar
- Besserer Kompromiss aus Platzbedarf und Redundanz
- Für VMs mit leichtem Anspruch akzeptabel
- 2x x-y Verlust an Platz für Redundanz

Raid 1+0

- Absolut notwendig für VM Einsatz mit drehenden HDDs, sonst Latenz schlecht
- Weitere Spiegel sind jederzeit anfügbar
- Einzelne Spiegel können während des Betriebs ohne Redundanzverlust getauscht werden (wenn Slots frei)
- Tausch zweier Datenträger führt zur Erweiterung der Kapazitäten (`zpool set autoresize=on <rpool>`)
- Jedoch 50% Platzverlust für Redundanz

Raid5+0

- Kompromiss

Solidstate Disks

- Bei ZFS immer zu bevorzugen, da Latenz ca 2x langsamer als ohne ZFS
- Für HDD Systeme immer Cache Device einsetzen, dieses muss nicht redundant oder hotswappable sein. Ausfall unterbricht nicht den Betrieb
  - **SATA**
    - nur 6GBit/s halb-duplex (ca. 600 Megabyte/s)
    - bei onBoard Controller darauf achten, mit wie vielen PCIe Lanes und mit welcher PCIe Generation angebunden, ggf. Flaschenhals
    - günstig
    - Samsung Pro, Intel mit krummen Zahlen (960GB, 1920GB), z.B. DC-S4510
    - nicht immer Hot-Swap möglich
    - SSDs im Formfaktor 2,5 Zoll und M.2 gängig
  - **SAS**
    - 12GBit/s und voll-duplex (ca. 1,2 Gigabyte/s)
    - teurer, aber weniger Müll am Markt
    - Hersteller noch unklar
    - Hot-Swap
  - **NVMe**
    - Protokoll setzt nativ auf PCIe auf, Latenz und Overhead gegenüber den klassischen Protokollen (SATA, SAS) sehr gering
    - Vollduplex
    - Bandbreite abhängig von PCIe-Generation und Anzahl der angebundenen Lanes (z.B. mit PCIe 4.0 x4 sind etwa 7 Gigabyte/s möglich, mit PCIe 3.0 x4 etwa 3,5 Gigabyte/s)
    - **M.2 Formfaktor**
      - nicht verwechseln mit M.2 SATA SSDs, PCIe / NVMe muss in den Specs stehen
      - Sinnvoll für Cache oder LOG
      - Wenn durch Bios bootfähig kann man sich ein Notsystem darauf bauen
      - günstig
      - belegen keinen Slot
    - **U.2 Formfaktor (SFF-8639)**
      - Bekannte irische Band LOL
      - Einsatzbereich Server und Storage
      - Wenig Erfahrung mit ZFS
      - Hot-Swap
      - PCIe 3.0 x4 Anbindung, Abwärtskompatibel zu SAS und SATA, Mischbetrieb über ein Kabel mit entsprechender Backplane möglich
      - teuer
    - **U.3 Formfaktor (SFF-TA-1001)**
      - Nachfolger der U.2 Schnittstelle
      - PCIe4.0 Unterstützung

- weitere Eigenschaften identisch zur U.2 Schnittstelle

RAIDZ-BUG

NAME	VOLBLOCK	LREFER	REFER
pool1/test5	8K	16.7G	35.7G (213% overhead)
pool1/test2	16K	89.4G	95.6G (6,93% overhead
pool1/test3	32K	89.9G	96.0G (6,78% overhead)
pool1/test4	64K	50.1G	53.5G (6,78% overhead)
pool1/test	128K	60.7G	60.7G (0% overhead)

ZFS Optimierung und Todsünden

ZFS Funktionen für SSDs

- `zpool get autotrim <poolname>`: Auslesen ob „autotrim“ eingeschaltet ist
- `zpool trim <poolname>`: Pool manuell trimmen. Nicht alle SSDs können das, aber die meisten
- `zpool status -t <poolname>`: Auslesen TRIM status
- `zpool set autotrim=on <poolname>`: generell trimmen (autotrim einschalten, wenn manuelles trimmen erfolgreich)
- `zfs set logbias=throughput <dataset/zvol>`: Eventuell bei großen Files spannend
- `zfs set sync=disabled  <dataset/zvol>`: kann Leistung steigern, auf Kosten der Sicherheit
- Es gibt weniger Müll an SAS SSDs als SATA am Markt
- Enterprise SSDs verwenden! 480G, 960G, 1920G, 3840G ....

ZFS mit Spindeln

- Mit mehr disks skalierst die Bandbreite
- Mit mehr vdevs skalierst IOps
- SAS (vollduplex) ist bei gleicher Bandbreite schneller als SATA (Halbduplex)!
- HBA muss den Linkspeed der Disk unterstützen, neuere Disk funktioniert an altem HBA und anders herum, Links Speed kann man u.a. mit lshw auslesen
- SATA Disks können an SAS Controller betrieben werden, anders herum nicht kompatibel

U.2 sticht SAS aus

- Viele Mainboards unterstützen SATA, SAS und U.2
- Direkte Anbindung an PCIe Bus, also kein 12G Limit

ZFS mit Cache und oder LOG

- `zpool add cache <poolname> diskxpx`: Eventuell Lesecache auf Boot NVMe vom PVE oder ganze Disk
- `zpool add log <poolname> **mirror** diskxpx  diskxpx`: Eventuell Logdevice auf Boot NVMe´**s** vom PVE oder ganze Disks im **Spiegel plus USV!**
- `zfs set sync=always <poolname>`: ohne Sync, kein Log

Todsünden

> **HW RAID Controller nutzen und mehrere LUNs als Raid VDEVs nutzen**
> Datenverust, kein Smart, keine Metriken

> **Single VDEV Raid z. B. auf Hardware Raid oder iSCSI/FC SAN**
> Bei Beschädigung kann keine weitere Disk die Redundanz, bzw. die Konsistenz bestätigen. Wenn ein VDEV zu viele Fehler liefert, lehnt ZFS es für einen weiteren Import dauerhaft ab!!! Eventuell zpool import -fFX poolname probieren, destruktiv für Rollback auf letzten guten Zustand, sehr riskant

> **Deduplizierung aktivieren**
> kann nicht mehr ausgeschaltet werden, ggf. in neueren ZFS Versionen optimiert

> **ZFS_ARC zu klein / zu groß**
> schlechte Leistung, bei Cache und LOG mehr ARC notwendig

> **Pool über 80% Füllstand**
> Leistungseinschränkung bis Stillstand bei ca 96%

> **Snapshot Bereinigung / Management aus dem Blick verlieren**
> führt ggf. unbeobachtet zu vollem Raid

> **Snapshots auf Backups**
> völlig kontraproduktiv, führt zu sehr vollem Raid

> **Anfügesnapshot durch schlechtes Management verlieren**
> Replikation benötigt neuen Platz und Zeit

Druckbetankung
dafür ist ZFS nicht gemacht, was mehr kann, braucht auch mehr Leistung

PVE auf selbem RAID wie Systeme (OS auf Datenpool)
Last auf VMs führt im Härtefall bis zum Kernelcrash

Kein ECC RAM
Datenbeschädigung die nicht in jedem Fall durch Scrub und Resilver behoben werden kann (selten)

Proxmox Datastore auf ZFS Dataset Directory setzen
Proxmox erstellt eventuell beim Booten den Ordner selbst und ZFS kann dann nicht mounten, Rollback setzt alle virtuellen Disks im Dataset zurück

Platzangaben vertrauen
Recordsizes, Volblocksizes, Compressions, Special Devices und Co. machen eine Vorhersage fast unmöglich und rechtfertigen Thin Provisioning

zpool upgrade zu früh oder zu spät
pool upgrade erweitert features die ggf. beim aktuellen Proxmox ISO noch nicht verfügbar, jedoch im Upgrade verfügbar sind. Im Recoveryfall kann man nur readonly mounten!

Festplatte ersetzen bei Proxmox ZFS Boot Systemen
zpool replace sollte nur nach Übernahme der Partitionstabelle einer funktionierenden Disk auf die dritte Partition ausgeführt werden, zusätzlich mit proxmox-boot-tool init/format auf die zweite Partition.

iSCSI LUNs mit Proxmox
nicht empfehlenswert, bestenfalls mit dem Free-/TrueNAS Plugin. Wie lange das supportet ist weiss keiner.

Import Proxmox Pool in Recovery Umgebung

Achtung Eigentümer des Pools ändert sich, daher zpool import -fa benutzen, beim nächsten boot wieder zpool import -fa und exit, /etc/pve ist in Recoveryumgebung nicht vorhande, da aus FUSE generiert

Billige USB Adapter und Gehäuse
Manche Adapter haben alle die gleiche Kennung, daher ggf. Raid erstellen und importieren über zpool import /dev/disk/by-path oder by-partuuid

Raid ohne dryrun erweitern
'zpool add **-n** rpool **mirror** sdc3 sdd3' wäre korrekt, wobei es tödlich wäre hier das wort mirror zu vergessen

Konsumer SSDs für Cache und LOG
Waren im Dauerbetrieb schon nach weniger als zwei Jahren am Limit, währen Enterprise SSDs nach 9 Jahren erst bei 30% waren. Getestet mit Samsung Consumer und Intel Enterprise

VMs nicht trimmen
Discard in VMs gibt mit SSD Emulation freie Blöcke an ZFS zurück, ansonsten sehr viel Platz verschenkt!

Startsektor der Partition nicht oktal

Manche Festplatten starten ab Block 34, manche auf Block 2048. Bei Startblock 34 sollte die erste Partition bis Block 2047 gehen, damit die nächste Partition direkt bei 2048 anfängt

Alles auf einmal machen
Betrieb, Scrub, Trim Backup und Replikation treiben die Load in die Höhe, nach fest kommt ab:)

Nur auf ZFS vertrauen
Auch wenn wir nun weit über 15 Jahre ZFS produktiv fahren, machen wir immer zusätzlich Replikate und Backups. Nur so kann wirklich nichts mehr passieren. Replikate bei denen die VM zum Snapshotten heruntergefahren

> wurde, dürften die absolut beste Sicherung sein und immer die erste Wahl bei Problemen

Update 2024 zum Thema Volblocksize

zfs create -o volblocksize=8k rpool/8k -V 1G

#Warning: volblocksize (8192) is less than the default minimum block size (16384).To reduce wasted space a volblocksize of 16384 is recommended.

rpool/16k **1.11G** 15.2T 81.4K -
rpool/8k **1.49G** 15.2T 81.4K -

NAME STATE READ WRITE CKSUM

rpool ONLINE 0 0 0

raidz1-0 ONLINE 0 0 0
scsi-20100000000000005cd2e475ae4b5651-part3 ONLINE 0 0 0
scsi-20100000000000005cd2e4f1bd4e5651-part3 ONLINE 0 0 0
scsi-20100000000000005cd2e4ef8f4b5651-part3 ONLINE 0 0 0
scsi-20100000000000005cd2e4889d4b5651-part3 ONLINE 0 0 0

Bei RaidZ2 wäre der Overhead über 200%

Inzwischen warnt ZFS, jedoch sieht das keine in der PVE GUI!

Emfpehlung Volblocksize für ZVOLs

Raid 1 / 10 8-16k

Raid5(Z1) 16k

Raid6+(Z2-3) 32k

Empfehlung Recordsize für Datasets

LXC 128k

Mailarchiv 16k

Proxmox Backup Server 1024k

ZFS Import - Importieren von ZPOOLS

Bei (Neu-)Installationen oder anderen Startproblemen z.B. von Proxmox ist es teilweise nötig, den *ZPOOL* manuell zu importieren (siehe auch **Proxmox Startproblem**).

Prüfen, ob es *zpools* gibt

```
zpool import
```

Importieren der *zpools*

```
zpool import -fa #Optionen -f force, -a all.
```

Die häufigste Ursache ist gerade bei mehreren internen Festplatten die Verzögerung.
Abhilfe schafft das Anpassen des mit einem Delay im *Grub-Loader*

```
nano /etc/default/grub
```

In dieser Konfigdatei wird folgende Zeile ergänz mit *rootdelay=10* (= 10 Sekunden Verzögerung)

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet rootdelay=10"
```

Wichtig: Danach immer GRUB aktualisieren mit

```
update-grub
```

bzw.

- `update-grub` *Aktualisierung der Grub Loaders*
- `grub-install /dev/sdb` *Installation des Grub Loaders auf weiteres Laufwerk (hier sdb)*
- `update-initramfs -u` *Aktualisiert die Grub-Infos auf diesen HDs*

Gibt es zwei identische //zpools// namens //rpool//, kann man diese folgender Weise importieren:

```
# Anzeigen der eindeutigen ID des zpools
zpool import
# Import des zpools per ID
zpool import -f <ID des zpools>
# Import des zweiten zpools mit identischen Originalnamen mit neuem Namen z.B. rpool2
zpool import <identischer zpoolname(org)> <neuer zpool name>
# Anzeigen des Mountpointes des zweiten zpools
zfs get mountpoint <neuer zpool name>
# Sollte dieser ebenfalls auf / (root) stehen in Proxmox wie folgt anpassen
zfs inherit mountpoint rpool/ROOT/pve-1
```

Wichtig: Der Mountpoint wird *IN DAS ZFS-FILESYSTEM* geschrieben, daher bei Änderungen oder Neuinstallationen von Proxmox mit mehreren *zpools* VORHER Haupt-zpool *rpool* umbenennen und Mountpoint anpassen!

Quelle: **Proxmox** und **Sysops-Matrix**

Import eines Pools mit temporärem Namen

Eindeutige ID anzeigen:

```
zpool import # z.B. id: 16760479674052375628
```

Import:

```
zpool import -R /a -t altrpool 16760479674052375628
```

```
# zpool list
NAME      SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
altrpool   97G  22.4G   74G  23%  1.00x  ONLINE  /a
rpool     465G  75.1G  390G  16%  1.00x  ONLINE  -
```

Quelle Oracle

ZFS Arc oder Arcsize

Arc, Arcstat oder Arcsize bezieht sich auf den *CACHE* im Arbeitspeicher zur Verwaltung eines *ZPOOLS*. Es ist eine Grundeinstellung eines ZPOOLs, nicht direkt eine Option eines Datasets.

Setzen des Cache Limit „Arcsize“

```
echo "options zfs zfs_arc_max=2147483648" >> /etc/modprobe.d/zfs.conf
```

oder manuell

```
nano /etc/modprobe.d/zfs.conf
2147483648
6442450944
```

Beispiele bzw. Berechnung

```
(8GB=8*1024*1024*1024)
```

Update der Eintragungen für ZFS

```
update-initramfs -u -k all
```

„echo 3 > /proc/sys/vm/drop_caches (??)

##ToDo: Erklärungen inkl. Abhängigkeiten von Cache-SSDs, Datengößen usw.

Neue Anzeigen der Eigenschaften der ZSF-Datasets

ZFS Options

```
zfs list # Anzeigen der Datasets
zfs list -t filesystem -o name,used # Anzeigen zusätzlicher Eigenschaften (hier //name// u
zfs get all <eines der datasets z.B. "rpool/data/vm-123-disk-0">
```

Anzeigen einer einzelnen Eigenschaft (z.B. Compression)

```
zfs get compression tank # nur vom ZPOOL tank
zfs get -r compression tank # Rekursive Anzeige aller
```

Setzen von Eigenschaften (2. Zeile Beispiel: Eigenschaft *com.sun:auto-snapshot* für Script *zfs-auto-snapshot*)

```
set compression=on tank/home/
zfs set com.sun:auto-snapshot=false tank/home
```

Setzen der Kompression auf alle unterliegenden *datasets*

```
zfs inherit compression tank
```

Überschreiben **aller** Eigenschaftswerte (Bitte beachten Sie, dass die Option -r die Eigenschaftswerte aller untergeordneten Datasets zurücksetzt.)

```
zfs inherit -r compression tank
```

SOURCE-Werte

SOURCE-Wert	Beschreibung
`default`	Dieser Eigenschaftswert wurde für dieses Dataset bzw. seine übergeordneten Datasets nie explizit gesetzt. Es wird der Standardwert für diese Eigenschaft verwendet.
`inherited` from *Dataset-Name*	Dieser Eigenschaftswert wurde vom übergeordneten Dataset geerbt, das in *dataset-name* angegeben ist.
`local`	Dieser Eigenschaftswert wurde mithilfe von `zfs set` für dieses Dataset explizit gesetzt.
`temporary`	Eigenschaftswert wurde mithilfe von `zfs mount - o` gesetzt und gilt nur solange, wie das Dateisystem eingehängt ist. Weitere Informationen zu temporären Eigenschaften von Einhängepunkten finden Sie unter **Verwenden temporärer Einhängepunkte.**
- (keiner)	Diese Eigenschaft ist schreibgeschützt. Ihr Wert wird von ZFS bereitgestellt.

Quelle **Doc.Orcale.com**

HD-Identifikation unter ZFS

- `zpool status` zeigt erst mal alle Platten und Datasets
- Unter */dev/disk/by-id/* sind die Festplatten per ID aufgelistet
- `by-id`
  - `ata-` zeigt klassischen S-ATA Platte mit Seriennummer
  - `wwn-` zeigt SAS-Platte (Nummer ebenfalls auf HD zu finden)
  - `usb-` USB Stick oder Festplatte
  - `iscsi-` iSCSI Platte
- `by-label`
  - zeigt die *rpools* (Daten/rpool/rpool2/System-reserviert) mit Mountpoints
- `by-partlabel`
  - `zfs-… → ../../sd(n)1` ZFS Symbolic-Links zu der Hauptpartition (1) der Disks sd(n)
- `by-uuid`
  - zeigt UUIDs für jede einzelne Partition inkl. Mountpoints *sd(nm)*
- `by-path`
  - gibt Aufschluss über die jeweilige Schnittstellen-Pfad (z.B. `pci-0000:44:00.0-sas-phy3-lun-0`)
- `lsblk -f` für Mountpoints
- `ls -althr` zeigt alle HDs mit ID an

Aktualisieren von ZFS-Speicher-Pools

Gerade nach Updates von *Proxmox* sollte man immer die *ZPOOL* prüfen und mit Update des ZFS-Systems abschließen. Eine ausführliche Erklärung gibt es im **Oracle Solaris ZFS-Administrationshandbuch**.

Um zu sehen, ob ein Upgrade nötig ist:

```
zpool status
```

Genauere Infos über den aktuelle *ZPOOL*

```
zpool upgrade -v
```

Um letztendlich alle (`-a`) *ZPOOLs* zu aktualisieren:

```
zpool upgrade -a
```

Dateiwiederherstellung von ZVOLs in Proxmox

- Voraussetzung: Backups über *zfs-auto-snapshot*
- `zfs clone rpool/data…/vm-<vmid>-disk-0@…auto-snapshot rpool/data/…/vm-<vmid>-disk-9`
- `cfdisk /dev/zd<nr>`
- `mkdir /mnt/restore`
- `mount /dev/zd<nr>p3 /mnt/restore/`
- `ls /mnt/restore/` zum Kontrollieren und Auslesen der alten Dateien und eigenen Einstellungen

Kennzeichnung der geklonten Disk mit Ziffer **9**
als Standard für geclonte Disks
ermöglicht Routine für Notfälle!

Löschen des Clones bzw. Rückgängigmachen der Schritte

- `cfdisk /dev/zd`
- `unmount /mnt/restore`
- `zfs destroy -r rpool/data/…/vm-<vmid>-disk-9`

Sharing unter ZFS - SAN Protkolle

Inhalt

- Übersicht Sharing (Samba, NFS, iSCSI oder SSHFS)
- NFS Sharing
- Samba Sharing
- iSCSI Sharing (unvollständig!)
- Vergleich SAN Protokolle
- Vergleich Fileserver Windows, UCS, TrueNAS und Zamba

Übersicht Sharing (Samba, NFS, iSCSI oder SSHFS)

- **Samba**
  - Schnittstelle zum Anwender
  - Für Filesharing
  - Kompatibel mit allen Betriebsystemen
  - Authentifizierung lokal oder AD
  - Verschlüsselt
  - Kein Routing sinnvoll möglich über NAT
- **NFS**
  - Schnittstelle zu iX Systemen (Linux, Mac, etc.)
  - Einfachste Konfiguration und Absicherung (/etc/exports)
  - Für Filesharing
  - Schnell
  - Einfach zu Mounten
- **iSCSI**
  - Komplex
  - Für Blocksharing von u. a. Disks, LVM, ZVOLS
  - Multipath
  - HA+RR etc
- **SSHFS**
  - Filesharing via SSH
  - Langsam
  - Für ESXi ohne vCenter zur Migration

NFS Sharing

Erzeugen Dataset und Mount unter */rpool/nfsshare*, ideal für *VMs* (raw, vmdk, qcpow2) oder *Files*:

```
zfs create rpool/nfsshare
```

Freigabe der Dateien:

```
zfs set sharenfs=on rpool/nfsshare
```

oder mit Zugriffsrechten aus dem Netz *192.168.11.0/24*:

```
zfs set sharenfs="rw=@192.168.11.0/24" rpool/nfsshare
```

Zugriffssteuerung

Vereinfacht für kompletten Share mit

```
chmod oder chown # nach Bedarf
```

Zugriffsteuerung per *Linux PAM* Benutzer mit Editieren der Datei `/etc/exports`

```
# freigabe1 wird für zwei Rechner freigegeben
# notebook darf nur lesen (ro)
# desktop darf lesen und schreiben (rw)
rpool/nfsshare     notebook(ro,async) desktop(rw,async)
```

oder kombiniert mit *IPs* oder *Netzwerken*

```
# Freigabe gilt nur für 192.168.1.13, jedoch nur mit Leserechten:
rpool/nfsshare     192.168.1.13(ro,async)
# Freigabe gilt für alle IPs von 192.168.1.1 bis 192.168.1.255, mit Lese-/Schreibrechten:
rpool/nfsshare       192.168.1.0/255.255.255.0(rw,async)
# Freigabe gilt nur für den Rechner mit dem Namen notebook
rpool/nfsshare     notebook(ro,async)
```

Bei Leistungsproblemen mit *NFS* kann man ggfs. *sync* deaktivieren mit `zfs set sync=disabled rpool/nfsshare`.

Weitere Infos zu den Risiken gibt es bei **https://jrs-s.net/2019/07/20/zfs-set-syncdisabled/**.

Mounten von NFS Shares

- Mount in Proxmox
  - *Datacenter* > *Storage* > Taste *Add* > NFS
  - ID: Mount benennen „myNAS1“
  - Server: IP Adresse des Server
  - Export: Auswahl der NFS Freigabe (taucht auf bei erfolgreicher NFS-Verbindung)
- VMware per Datastore Dialog
- HyperV über Erweiterung Windows Funktionen
- Linux
  - `mkdir /mnt/nfsmount && mount -t nfs 192.168.0.100:/rpool/nfsshare /mnt/nfsshare`
  - oder
  - per fstab
    - `192.168.0.100:/rpool/nfsshare /mnt/nfsshare nfs rw 0 0`
  - wenn der umount mal scheitert:
    - `umount -l /mnt/nfsshare #lazy umount`

Samba Sharing

Erzeugen Dataset und Mount unter */rpool/smbshare*:

```
zfs create rpool/smbshare
```

Freigabe der Dateien:

```
zfs set sharesmb=on rpool/smbshare
chmod 777 /rpool/smbshare
```

Zugriffssteuerung

Vereinfachtes Sharing mit

```
smbpasswd -a root    # erzeugt Samba Passwort
chmod oder chown     # nach Bedarf
```

Detailliertere Zugriffsteuerung mit Editieren der Datei `/etc/samba/smb.conf`

```
[Global]
workgroup = workgroup security = user
map to guest = Bad Password

[homes]
comment = Home
Directories browsable = no
read only = no
create mode = 0777

[smbshare]
path = /rpool/smbshare
public = yes
writeable = yes
comment = smb
share printable = no
guest ok = yes
create mask = 0777
security mask = 0777
directory mask = 0777
force create mode = 0777
directory security mask = 0777
```

Kontrolle der effektiven Config `testparm` (zum Kontrollieren der effektiven Konfiguration) und Abschluss mit

```
service smbd restart
```

Zugriff auf SMB Freigaben

- Windows: `\\<IP>\smbshare`
- macOS und Linux: `smb://<IP>/smbshare`
- Nötige Erweiterung für Linux-Zugriff
  - ```
    apt-get install cifs-utils
    mkdir /mnt/smbshare
    mount -t cifs //ip/smbshare /mnt/smbshare #username=root
    ```
  - per fstab
    - `ip/smbshare /mnt/smbshare cifs iocharset=utf8 0 0`

hoch

iSCSI Sharing

Unvollständig

Erzeugen Dataset und Mount unter */rpool/iscsishare*:

```
zfs create rpool/iscsishare
```

Freigabe der Dateien:

???

Zugriffssteuerung

Vereinfachtes Sharing mit

???
```

Detailliertere Zugriffsteuerung mit Editieren der Datei `/etc/???`

Zugriff auf SMB Freigaben

- Windows:
- Proxmox:
- macOS und Linux:
- Nötige Erweiterung für Linux-Zugriff
  - ```
    ???
    ```
  - per fstab
    - ???

Vergleich SAN Protokolle

- **Vorteile NFS**
  - schlank
  - einfach
  - übersichlich bei Fehlersuche
  - schnell
- **Nachteile NFS**
  - Wenig Funktionen
  - kein Multipath (Gleichzeitiger Zugriff von mehreren Servern)
  - kein Sharing von Volumes
  - keine UUID (eindeutige Identifizierung)
- **Vorteile iSCSI**
  - Multipath
  - HA
  - Sharing von Volumes
  - Blockgrößen und etliche Paramter
  - Eindeutige Identifizierung
  - Hoher Funktionsumfang
- **Nachteile iSCSI**
  - aufwendiges Setup
  - schwierige Fehlersuche
  - schlechte Integration PVE
    ```

Vergleich Fileserver Windows, UCS, TrueNAS und Zamba

- **Windows**
  - Natives SMB
  - Kritischer Windows Unterbau
  - Partitionierung notwendig
  - NTFS Probleme
  - Kein ZFS, dafür wenige Wiederherstellungspunkte möglich
- **Univention**
  - Klassiches, aktuelles Samba
  - Umfangreiches System als Unterbau
  - LVM bei Kapaziätserweiterung notwenidg
  - EXT4
  - Kein ZFS, keine Schattenkopien
- **TrueNAS**
  - Volle GUI, ZFS und Replikationen, Monitoring
  - Extra Hardware
  - Läuft direkt auf Hardware, daher schnell
- **Zamba**
  - Schnelle Bereitstellung per LXC
  - Schlank und schnell, da quasi direkt auf ZFS
  - Schattenkopien so viele man möchte
  - ZFS GUI via Cockpit, Wiederherstellung via Windows-Explorer
  - Erste Wahl bei Einsatz von Debian oder Proxmox

hoch

— Andi Treml 2021/05/15 14:33-16:

Proxmox VM mit ZFS Auto-Snapshots klonen

Falls man bei Proxmox die Autosnapshots aktiviert hat, ist es recht einfach einen Klon oder von einem Replikat einen Snapshot zum Test zu booten, ggf. ohne Netzwerk

Auflisten der VMs

qm list

2001 OpnSense-Node-PVE-HA running 16384 40.00 3675016

Auflisten der Disks

zfs list | grep **2001**

*rpool-hdd/data/sysops/vm-**2001-disk-1***

Auflisten der Snapshots

zfs list -t snapshot rpool-hdd/data/sysops/vm-2001-disk-1

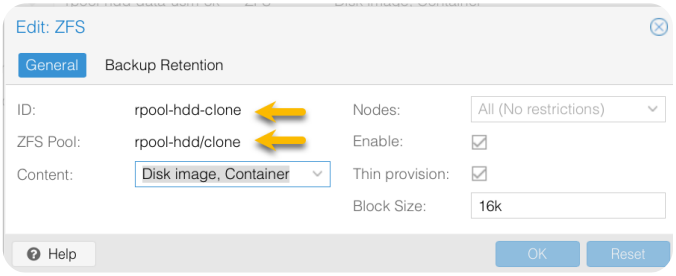
rpool-hdd/data/sysops/vm-2001-disk-1@backup-zfs_2023-05-04_01:04:24 15.9M - 13.8G -
rpool-hdd/data/sysops/vm-2001-disk-1@zfs-auto-snap_hourly-2023-05-03-2355 16.0M - 13.8G -
rpool-hdd/data/sysops/vm-2001-disk-1@zfs-auto-snap_hourly-2023-05-04-0055 22.6M - 13.8G -
rpool-hdd/data/sysops/vm-2001-disk-1@zfs-auto-snap_hourly-2023-05-04-0155 18.7M - 13.8G -
rpool-hdd/data/sysops/vm-2001-disk-1@zfs-auto-snap_hourly-2023-05-04-0255 18.8M - 13.8G -
...

Klonen der Disk, keine Last, kein Platz, keine Zeit

zfs clone rpool-hdd/data/sysops/vm-2001-disk-1@backup-zfs_2023-05-04_01:04:24 rpool-hdd/clone/vm-2001-disk-9

rpool-hdd/clone/vm-2001-disk-9 35.9M 11.3T 13.8G -
rpool-hdd/data/sysops/vm-2001-disk-1 21.5G 11.3T 13.8G -

Datastore anlegen für Clones



Konfig kopieren

cd /etc/pve/qemu-server #dort liegen die VM Definitionen des aktuellen Hosts
cp 2001.conf 9992001.conf #kopiert die Konfiguration, Achtung wegen Autostart und Nettwerkkarten!
nano 9992001.conf

Dort die Disk ändern auf **rpool-hdd-clone:vm-2001-disk-9 #Achtung: muss geändert werden, sonst bootet die Disk aus zwei VMs und geht defekt!**

Dann die VM 9992001 Netzwerkkarten in GUI optional deaktivieren

qm start 9992001

Am Ende kann die VM gelöscht werden, der Klon sollte verschwinden, ansonsten kann er über die GUI im Store rpool-hdd-clone gelöscht werden!

Sichere Prüfung zum Klon löschen

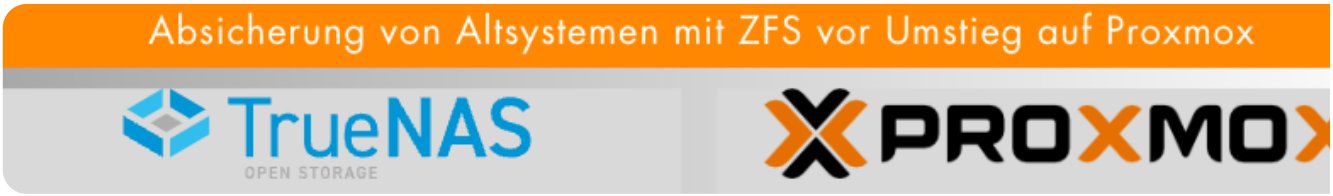
zfs get origin rpool-hdd/clone/vm-2001-disk-9 #nur ein Klon hat ein Origin

rpool-hdd/clone/vm-2001-disk-9 origin rpool-hdd/data/sysops/vm-2001-disk-1@backup-zfs_2023-05-04_01:04:24 -

zfs destroy -r rpool-hdd/clone/vm-2001-disk-9 #der ist dann definitiv für immer weg

cz

Schutz mit ZFS für VSphere und Hyper-V



TrueNAS Scale/Core

Es ist nicht notwendig auf teure, proprietäre SAN Systeme zurückzugreifen. Die Erfahrung zeigt dass gerade diese Geräte im Notfall eine hohe Abhängigkeit vom Hersteller mit sich bringen. Ebenfalls wird oft der Support verweigert wenn die SAN nicht den Vorstellungen des Hypervisor Supporters entspricht und umgekehrt. Mit einer TrueNAS SAN Lösung kann fast jede beliebige Server Hardware als Basis genutzt werden. Open ZFS ist in der Scale Version in Debian, in der ausgereifteren Core Version in FreeBSD enthalten und seit über 15 Jahren bewährt! Es entstehen keine Kosten für den Betrieb und die Updates gehen erfahrungsgemäß über eine Dekade ohne Probleme. Verschlüsselung, Replikation, Monitoring und Snapshots sind hier Standard. Ein Hardware Raid Controller wird wie immer nicht benötigt und man spart so noch mehr Geld!

Proxmox Virtual Environment

Ist ein Umstieg auf Proxmox geplant, profitiert man hier durch die hohe Flexibilität vom darunter liegenden Debian. Sowohl Hypervisor, also auch ZFS sind hier im Kernel enthalten. Für kleinere Installationen ist der NFS-Kernel-Server hier eine willkommene, zu VMware VSpehere sehr kompatible Alternative zum üblichen iSCSI Protokoll. So kann Proxmox hier für die Übergangszeit sowohl SAN, als auch später die Hypervisor Rolle übernehmen, nahtlos! Sobald die erste virtuelle Maschine auf Proxmox mit ZFS migriert wurde, profitiert sie sofort von Raid, Snapshots, Kompression und Replikationen. Eine sehr einfach umzusetzende, verständliche und nachhaltige Lösung!

Variante NAS auf ZFS mit TrueNAS oder Zamba aus der LXC-Toolbox

Liegt der Fokus eher auf Dateiablage und nicht auf virtuellen Maschinen, so kann man diese einfach auf ZFS ablegen. Der Anwender wird sofort unter der Explorer Funktion “Vorherige Versionen” alle Snapshots des Systems sehen. Active Directory Integration ist selbstverständlich vorhanden, wie auch die Administration der Benutzerrechte über den gewohnten Weg. Dank der eigens entwickelten Zamba Lösung ist es nicht mehr notwendig

dedizierte Hardware für ein ZFS basiertes NAS zu verwenden, ist aber trotzdem möglich!

Backup durch ZFS absichern

Wie bereits beschrieben können Backups aller Systeme auf ZFS NAS Lösungen gespeichert und tagelang mit Snapshots gegen Kompromittierung abgesichert werden. Als Hardware geht es hier von Raspberry Pi4 bis High End Server!

Hardware für ZFS

Sie finden spezielle Hardware für Proxmox, TrueNAS und Opensense bei unseren Partnern Thomas-Krenn.AG (neu) und Servershop24 (gebrauchte Geräte, bitte mit HBA anfragen)



zfs.rocks Videos bei YouTube

ZFS Ashift

Bei **ZFS** bezeichnet `ashift` die **Blockgröße (Sektorgröße)**, mit der ZFS **intern auf einem vdev arbeitet**.
Der Wert ist ein **Exponent zur Basis 2**:

```
Sektorgröße = 2^ashift Bytes
```

Bedeutung von ashift 9 bis 12

ashift	Sektorgröße	Entspricht
9	512 Byte	klassische 512B-Sektoren
10	1.024 Byte	1 KB
11	2.048 Byte	2 KB
12	4.096 Byte	**4 KB (Standard heute)**

Warum ashift wichtig ist

ZFS schreibt **immer in ashift-großen Blöcken** auf die Platte.

Wenn:

- die **physische Sektorgröße** der Platte größer ist als ashift
- oder das Storage intern mit größeren Blöcken arbeitet

→ entstehen **Read-Modify-Write-Zyklen**, was:

- Performance kostet
- unnötige Schreibvorgänge erzeugt
- SSDs schneller altern lässt

Praxisbezug (heute sehr wichtig)

Moderne Festplatten & SSDs

- Fast alle **HDDs**: 4K physische Sektoren (Advanced Format)
- **SSDs / NVMe**: intern ebenfalls ≥ 4K

ashift=12 ist heute der sichere Standard

Typische Szenarien

ashift=9 (512B)

Nicht mehr empfehlenswert

- schlecht für moderne HDDs/SSDs
- Performanceeinbrüche möglich
- kann nicht nachträglich geändert werden

ashift=12 (4K)

Best Practice

- kompatibel mit:
  - HDDs
  - SSDs
  - NVMe
  - RAID-Controllern
- minimaler Platzverlust
- stabile Performance

ashift > 12 (z. B. 13 = 8K)

⚠ Nur sinnvoll bei:

- speziellen Enterprise-Systemen
- dediziertem Storage mit garantierter Blockgröße

Wichtig: ashift ist **final**

- Wird **beim Erstellen des vdevs festgelegt**
- **Nicht änderbar**
- Auch beim Ersetzen einzelner Platten bleibt ashift gleich

□□ Fehlerhafte Wahl = Pool neu erstellen

Empfehlung (kurz & klar)

Einsatz	Empfehlung
Heimserver / NAS	**ashift=12**
SSD / NVMe	**ashift=12**
Unbekannte Hardware	**ashift=12**
Alte 512B-Platten	trotzdem **ashift=12**

Prüfen, welchen ashift ein Pool hat

```
zdb -C poolname | grep ashift
```

Wenn du möchtest, erkläre ich dir auch:

- **Zusammenspiel von ashift & recordsize**
- **Einfluss auf ZVOLs / VM-Storage**
- **ashift bei RAIDZ vs. Mirror**
- **Performance-Tuning für SSD-Pools**