

# Sharing unter ZFS - SAN Protokolle

## 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 Betriebssystemen
  - 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, qcow2) 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
- übersichtlich 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 Parameter
- 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**

- Klassisches, aktuelles Samba
- Umfangreiches System als Unterbau
- LVM bei Kapazitätserweiterung notwendig

- 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 Tremel 2021/05/15 14:33-16:

---

Version #2

Erstellt: 4 Januar 2023 21:47:44 von Admin

Zuletzt aktualisiert: 13 September 2023 12:54:42 von Admin