TrueNas VM + RustFS

Suggested setup

1. TrueNAS VM: Pass the disks or, preferably, their controller through to TrueNAS so ZFS sees the actual drives. Don’t create the pool from virtual disks if you can avoid it; TrueNAS warns that virtualized storage for regular or critical data is not a supported setup and may risk corruption. truenas.com
2. Make separate datasets: For example, `pool/apps-nfs` and `pool/rustfs-data`. Export them over NFS, restrict the exports to your app/RustFS VM addresses, and keep permissions deliberate.
3. RustFS VM: Keep its OS and VM disk on Proxmox’s local storage; mount `pool/rustfs-data` inside the VM and configure RustFS to use that mount as its data directory. RustFS supports single-node operation and an S3-compatible API.
4. Apps: Point Ente and PeerTube to RustFS’s S3 endpoint. Make that endpoint reachable from both the app/server side and the clients that upload directly—Ente’s presigned uploads require both to reach the same address. PeerTube’s S3 setup also requires attention to public access and CORS for video delivery.
5. Other apps: Mount `pool/apps-nfs` directly from TrueNAS where a regular shared filesystem is appropriate.

I’d avoid using the same NFS export for Proxmox VM disks and for application/object data. If you do want NAS-backed VM disks, make a separate dataset/export for that. NFS can work for Proxmox VM storage, but it adds network and NAS availability as dependencies for the VMs; keeping their disks local is simpler for this design.

Important tradeoff: the RustFS data is still on the mirror, so it is protected against one disk failing, but the mirror is not a backup. Take separate backups of important photos and videos. Also, with TrueNAS and RustFS as separate VMs, RustFS depends on TrueNAS being up and its NFS mount being available. A single mirrored pair does not provide storage high availability.

Revision #1
Created 2026-10-27 11:34:25 UTC by dpasn
Updated 2026-10-27 11:34:38 UTC by dpasn