UnRAID Operational Gotchas

Overview

Accumulated friction points from running UnRAID on Lusankya and integrating it with Centerpoint's Docker stack. Each entry includes root cause and the working fix.

1. NFS "Stale File Handle" After Mover Runs

Symptom: A client (Centerpoint) holding an NFS mount to a Lusankya share suddenly gets `Stale file handle` on `ls` or file access, requiring an unmount/remount to recover.

Root cause: When Mover relocates a file between cache and array, the underlying FUSE file ID changes. NFS clients cache file handles tied to the old fileid; UnRAID's `fuse_remember` tunable controls how long those handles are cached client-side, and a mismatch after a Mover-triggered fileid change produces "stale handle" until the client refreshes.

Fix:

- Force NFS **v4.2** on the client mount (NFSv3 is far more prone to this than v4.x).
- In `/etc/fstab` on the client, use `_netdev,nofail,x-systemd.automount,x-systemd.idle-timeout=300` rather than a static mount — automount re-establishes the handle on next access instead of hanging on a dead one.
- On the UnRAID side, the `fuse_remember` tunable (Settings → Global Share Settings) can be tuned if this happens frequently outside Mover events — but automount on the client side resolves it without touching server config.

2. Centerpoint Mounts Not Loading After Restart

Symptom: After a reboot, none of Centerpoint's NFS/CIFS mounts to Lusankya come back automatically; Docker containers depending on them fail to start.

Root cause: Docker services start before network mounts are ready at boot — a race condition, not a Lusankya-side fault.

Fix: `sudo mount -a` recovers immediately. To prevent recurrence, add `_netdev` to the relevant `/etc/fstab` lines so systemd waits for network availability before attempting the mount, and consider `x-systemd.automount` for more graceful boot-time handling.

3. Portainer 500 Error on Compose Redeploy After Changing NFS Path

Symptom: Updating a stack's NFS source path (new IP or mount point) in Portainer fails with a generic `Request failed with error code 500`.

Root cause: Docker refuses to redefine an existing **named volume's** `driver_opts` in place — the old volume definition with the stale NFS config is still registered.

Fix: Either `docker volume rm` the stale named volumes before redeploying, or — the more durable fix — replace named NFS volumes with direct bind mounts to an already-working host-level mount (`/mnt/data/...`) per the established Centerpoint convention of direct bind mounts over named volumes with `driver_opts`.

4. Immich / Any App Expecting Sentinel Files on a Pre-Existing Library Path

Symptom: An app (e.g. Immich) crashes on startup when its data volume points at a pre-existing directory rather than an empty one — it's looking for first-run sentinel files (`.immich`, etc.) that only get created during fresh initialization.

Fix: Manually create the expected subdirectories and sentinel files before first container start, with ownership matching the container's expected UID (commonly 1000).

5. UnRAID Array ≠ Traditional RAID 6

Clarification, not a bug: UnRAID's array uses per-disk XFS/BTRFS filesystems with a dedicated parity calculation layer — single parity drive here means tolerance for **one** failed data disk, recovered by rebuilding from the remaining disks + parity. This is functionally different from striped RAID 6 (which tolerates two failures by design). Don't assume RAID-6-equivalent fault tolerance when planning around this array; a second parity drive would be required to match that.

6. Cache-Only Shares Have No Mover Fallback

Symptom: None directly observed yet on this host, but worth flagging — `appdata`/`docker`/`system` are configured `shareUseCache="only"`. If the 2-disk BTRFS cache pool fills or both SSDs fail, there's no array fallback; writes simply fail.

Mitigation: Monitor cache pool free space proactively; don't let it run consistently above ~85% utilized.

7. Boot Flash / USB Enclosures Don't Report Standard SMART

Symptom: Diagnostics SMART reports for the boot flash drive and any USB-bridged drive come back empty — easy to misread as a tool failure.

Root cause: Most USB-to-SATA bridges and flash media don't pass through SMART attributes the same way native SATA/SAS does. This is expected, not a fault — but it does mean those devices have **no early-warning health signal**, so physical inspection / replacement-on-schedule is the only mitigation for the boot flash itself.

Last Updated

2026-06-23

Revision #1
Created 2026-06-23 22:11:47 UTC by Admin
Updated 2026-06-23 22:12:29 UTC by Admin