

Configure local VM and backup storage in Proxmox VE

Prepare additional local disks on a Proxmox VE node and give each one a distinct role: an LVM-thin pool for VM and container disks, an ext4 directory for local backups, and a spare held for future needs, then prove the layout with a test backup and restore.

PROXMOX VE 9.2 · LVM-THIN · EXT4 BACKUP DIRECTORY · RESTORE-TESTED · HOW-TO GUIDE

General educational guidance, not affiliated with or endorsed by Proxmox Server Solutions GmbH. The wipe and create steps destroy data on the disks they touch: identify every disk by model and serial number, and back up anything you still need first. Validated on Proxmox VE 9.2.10. Published 2026-09-08.

3 disk roles

fast VM pool · local backup target · documented spare

9 setup steps

identify, wipe, create both storages, restrict content, then prove it with a restore

A Proxmox VE virtualization host often arrives with extra disks that still carry partitions and old data from earlier use. This guide takes those disks from unknown state to assigned roles: a fast LVM-thin pool for virtual machine (VM) and container disks, a separate ext4 directory for local backup archives, and content restrictions that stop Proxmox from placing backups on the system disk or workloads on the backup disk. You identify every disk by device path, capacity, model, and serial number before any destructive step, and you prove the finished layout with a test VM, a backup, and a restore.

CONTEXT

Where this applies

A standalone node with locally attached disks

LVM-thin pools and directory storage are node-local: a cluster does not share them between nodes automatically. The same choices apply to a node inside a cluster; the steps here assume one node.

The node was installed on the LVM default layout

The steps assume the installer created `local` on `/var/lib/vz` and `local-lvm` on the system disk. A node installed on ZFS has `local-zfs` instead, and one installed on BTRFS has `local-btrfs` with `local` disabled. The disk work is the same; the storage names in steps 06 and 07 are not.

The permissions the work needs

In `pve-storage` 9.1.8, reading the disk list needs `Sys.Audit` on `/`, and the disk and storage work needs `Sys.Modify` on `/` plus `Datastore.Allocate` on `/storage` for **Add Storage**. **Wipe Disk** declares no permission at all, which Proxmox treats as `root@pam` only, so no role covers step 02. The validated example used `root@pam`.

The target disks are approved for destruction

Existing data on the selected additional disks is no longer required, or has been backed up and the backup verified. Each target disk shows up as an individual block device under **Disks > Disks** on the node. Where organizational policy requires it, obtain a maintenance window and change approval first.

The device names here are examples

Linux device names such as `/dev/sda` and `/dev/nvme1n1` are not universal, and they can change when hardware or controllers change. Replace every example name and value with ones verified on the target system.

MAP The validated layout

The example server held four SSDs. The system disk was recognizable by its existing BIOS boot, EFI, and LVM partitions, and it was never wiped, initialized, or otherwise modified.

DEVICE	MEDIA	CAPACITY	ROLE	RESULTING STORAGE
<code>/dev/nvme0n1</code>	Intel NVMe SSD	512 GB	Existing Proxmox system disk	<code>local</code> and <code>local-lvm</code>
<code>/dev/nvme1n1</code>	Seagate FireCuda 520 NVMe SSD	2 TB	Primary VM and container storage	<code>vmfast</code> (LVM-thin)
<code>/dev/sdb</code>	Crucial MX500 SATA SSD	2 TB	Local backup target	<code>backup-local</code> (ext4 directory)
<code>/dev/sda</code>	Crucial MX200 SATA SSD	1 TB	Unused spare	None

The named SSD models document the validated example only. For client or production use, select server-qualified media whose endurance, power-loss behavior, interface, and vendor support match the workload and recovery requirements.

STORAGE ID	BACKEND	PERMITTED CONTENT	PURPOSE
<code>local</code>	Directory at <code>/var/lib/vz</code>	ISO image, Container template (Import optional)	Installation media and templates on the system disk
<code>local-lvm</code>	LVM-thin	Disk image, Container	Existing system-disk thin pool, retained as secondary local capacity
<code>vmfast</code>	LVM-thin	Disk image, Container	Primary VM and container disk storage
<code>backup-local</code>	Directory at <code>/mnt/pve/backup-local</code>	Backup	Dedicated local Proxmox backup target

All four storage definitions are local to the node, enabled, and never marked as shared.

WHY Design decisions

LVM-thin for the VM pool

SPACE ALLOCATED AS GUESTS WRITE • SNAPSHOTS AND CLONES

LVM-thin (thin provisioning on the Linux volume manager) allocates physical blocks only as guest data is written, and it gives Proxmox efficient snapshot and clone support. A newly created 200 GB virtual disk consumes almost nothing at first; its physical footprint grows as the guest writes. A thin pool that runs out of data or metadata space stops guest writes, and the pool itself may be left damaged and hard to repair, so watch both figures and do not overcommit capacity without monitoring and a documented response. Choose regular LVM instead when every volume must have its full space reserved up front.

A plain ext4 directory for backups

VZDUMP ARCHIVES ARE FILES • CONTENT LOCKED TO BACKUP

Standard Proxmox backups (vzdump archives) are files, and a directory backend on ext4 is a straightforward local filesystem for Proxmox to store them on. The backup disk is dedicated to the **Backup** content type, which stops new VM disks, containers, ISO files, or templates from landing there by accident.

Backups live on their own disk

A BACKUP ON THE VM'S OWN DISK FAILS WITH IT

A backup stored on the same physical disk as the running VM cannot survive that disk failing. Keeping the VM pool and the local backup filesystem on separate disks means a dead VM disk still leaves the backups and the host to restore from. This layer is local only: it cannot protect against total host loss, theft, fire, a destructive power event, or an administrative action that affects the whole server. Critical systems need an off-host copy as well; the operations section covers what that takes.

Neither data disk is mirrored

CAPACITY OVER REDUNDANCY • A DISK FAILURE MEANS A RESTORE

This layout spends both disks on capacity rather than redundancy. If the VM pool disk fails, every guest on it is gone until it is restored, so the recovery point is the last backup and the recovery time is a restore. If the backup disk fails, the guests keep running and the local restore points do not. Decide whether that is acceptable before creating any storage, because the alternative below changes the disk layout.

Why ZFS was not selected here

UNEQUAL DISKS • THE MIRROR WOULD CONSUME THE BACKUP DISK

ZFS (a combined filesystem and volume manager with built-in checksums, snapshots, and compression) is a valid Proxmox option. It fit the example's unequal disks and intended roles poorly.

Mirroring the 2 TB NVMe SSD with the 2 TB SATA SSD would yield about 2 TB usable before overhead, writes would run at the SATA disk's speed, and the mirror would consume the disk meant to be the backup target. A three-disk RAIDZ1 (one disk's worth of capacity spent on parity, so the pool survives one disk failing) across 2 TB, 2 TB, and 1 TB is constrained by the smallest member, so it also gives about 2 TB usable and leaves the extra space on the two 2 TB disks unused.

A single-disk ZFS pool can detect corruption and offer snapshots, and without a second data copy it cannot repair ordinary corruption. If keeping the host running through a disk failure matters more to you than a separate local backup target, a ZFS mirror is a reasonable choice instead; decide before creating LVM or directory storage, because changing the design later means migration and disk reconfiguration.

WARN Before any destructive step

Wipe Disk, **Initialize Disk with GPT**, **Create: Thinpool**, and **Create: Directory** all modify disk metadata, and storage creation partitions or formats the selected disk. Selecting the wrong disk can make the Proxmox host unbootable or destroy client data.

- **Identify every disk positively, then record the mapping**

Match each target by device path, capacity, model, and serial number, and identify the Proxmox system disk explicitly before touching anything. A disk's current `/dev/sdX` name, its position in a list, or a visual selection highlight is never sufficient identification on its own. Record the approved disk-to-role mapping in the change record.

- **Confirm the backup, and confirm nothing still uses the disk**

Before wiping, confirm the required data is backed up, confirm the backup opens, and confirm no VM, container, mount point, volume group, ZFS pool, Ceph service, or other application still uses the target disk.

- **What Wipe Disk does**

In `pve-storage 9.1.8`, **Wipe Disk** runs `wipefs --all` against the selected device and its detected child partitions, then writes zeros over the first 200 MiB of the device (or the entire device when it is smaller than 200 MiB). Every other block keeps its old contents, so treat the operation as signature removal rather than media sanitization, and follow the organization's approved secure-erasure or destruction standard when disposing of media.

01 Inventory the disks and identify the system disk

Sign in to the Proxmox web interface, select the node (the example uses `pve01`), and open **Disks > Disks**. For every physical disk, record the device path, capacity, model, serial number, current usage, partition layout, GPT status, and S.M.A.R.T. status (the drive's built-in self-monitoring). Identify the Proxmox system disk and mark it as excluded from the change; in the validated example it was `/dev/nvme0n1`, carrying the BIOS boot, EFI, and LVM partitions.

Cross-check from the node shell. For destructive changes, run a second identification pass from **Shell** on the node:

```
lsblk -e 7 -o NAME,PATH,SIZE,MODEL,SERIAL,TYPE,FSTYPE,MOUNTPOINTS
pvs -o pv_name,vg_name,pv_size,pv_free
findmnt /
findmnt --target /var/lib/vz
```

- `lsblk` shows device ancestry, filesystems, models, serial numbers, and mount points.
- `pvs` shows which disks or partitions are LVM physical volumes and the volume groups they belong to.
- The two `findmnt` commands show what backs the root filesystem and which filesystem holds `/var/lib/vz`.

If the GUI and the shell disagree, stop and resolve the discrepancy before changing anything.

Review S.M.A.R.T. health. Select each intended target disk and choose **Show S.M.A.R.T. values**. Confirm the overall status passes and review the health, error, and Wearout indicators. A passing status is no guarantee against failure: investigate abnormal error counts, critical warnings, rapidly increasing media errors, or heavy wear before assigning a disk to production work.

02 Wipe the approved data disks

Complete this step only for disks approved for data destruction, one disk at a time, in **Disks > Disks**:

- Select the first approved data disk and reconfirm its capacity, model, and serial number against the change record.
- Select **Wipe Disk**, review the confirmation prompt, and complete the operation.
- Wait for the task to finish successfully, then repeat separately for each additional approved disk.
- Never select the Proxmox system disk.

- **Wipe Disk fails with a mounted or holder error**

Proxmox leaves the button enabled whenever a disk row is selected and refuses the operation on the node instead, reporting that a partition of the disk is mounted or that the kernel shows a holder such as an active LVM volume. Identify the dependency and remove it through the appropriate storage-management process first; bypassing the condition blindly risks an active volume group or mounted filesystem.

Those two checks are the whole guard, and neither one sees a disk that belongs to an imported ZFS pool, so on a ZFS-installed node confirm the serial number before selecting anything.

03 Initialize with GPT and pause at the checkpoint

Neither storage workflow requires a partition table, and the LVM-thin workflow removes the one you make here. Initialize anyway, because it gives every data disk a known, checkable state before the destructive work starts, and it leaves the spare disk in a state you can document. Still in **Disks > Disks**, select each approved blank disk, choose **Initialize Disk with GPT** (GPT is the GUID Partition Table, the modern partition-table format), and confirm. Select **Reload** if the display does not refresh on its own. Each prepared data disk should now show:

COLUMN	EXPECTED VALUE
Usage	No
GPT	Yes
S.M.A.R.T.	PASSED, where supported

Stop here and confirm that GPT exists on each data disk and that no Proxmox storage has been created yet. The storage workflows will then add or replace disk metadata. The Directory workflow creates a Linux filesystem partition when given a whole disk, and the LVM-thin workflow initializes the selected device as an LVM physical volume, so review the final layout after storage creation rather than assuming the pre-creation GPT state survives.

- **The spare disk stays empty on purpose**

The 1 TB `/dev/sda` stayed initialized with GPT and otherwise unused (**Usage: No, GPT: Yes**). Leaving the disk unassigned keeps it available for: a secondary backup target for selected critical workloads, temporary migration or recovery capacity, ISO or import or archive storage, or replacement capacity while another disk is serviced. Do not aggregate unrelated disks with RAID0 or a linear/JBOD span (concatenating disks into one volume) merely to present a larger one: a single member failure can take the combined storage down and may compromise everything in the aggregate.

04 Create the vmfast thin pool

The validated example assigned the 2 TB NVMe SSD at `/dev/nvme1n1` to primary VM and container storage. On the node, open **Disks > LVM-Thin** and select **Create: Thinpool**:

FIELD	VALIDATED VALUE
Disk	<code>/dev/nvme1n1</code> , verified against the approved disk inventory
Name	<code>vmfast</code>
Add Storage	Enabled, which creates the matching Proxmox storage definition automatically

Reconfirm the selected device is the intended one, select **Create**, and wait for the task to complete. Then verify in **Disks > LVM-Thin**:

- A thin pool named `vmfast` exists, in volume group `vmfast`.
- Its size is the disk capacity less the pool's metadata volume and the spare copy LVM keeps of it. The Proxmox interface also reports sizes in binary units, so the validated 2 TB disk read as about 1.82 TiB before the pool was created, and the pool itself as about 1.79 TiB. Both figures are expected rather than missing capacity.
- Initial data usage is at or near 0%, and metadata usage is low. A small amount of initialized metadata is normal.

- **Leave the existing pool named data alone**

The thin pool `data` in volume group `pve` belongs to the default `local-lvm` storage on the Proxmox system disk. Removing or repurposing it is a separate redesign with its own approval, never a side effect of this change.

In **Datcenter > Storage**, confirm the new definition: ID `vmfast`, type LVM-Thin, content Disk image and Container, shared No, enabled Yes.

05 Create backup-local and lock it to backups

The validated example assigned the 2 TB SATA SSD at `/dev/sdb` to local backups. On the node, open **Disks > Directory** and select **Create: Directory**:

FIELD	VALIDATED VALUE
Disk	<code>/dev/sdb</code> , verified against the approved disk inventory
Filesystem	<code>ext4</code>
Name	<code>backup-local</code>
Add Storage	Enabled

Reconfirm the target disk, select **Create**, and wait for the task. The workflow creates the filesystem and the mount configuration; in the validated system the filesystem mounted at `/mnt/pve/backup-local`, and the **Disks > Directory** view showed the underlying device through a persistent `/dev/disk/by-uuid/` path.

- **The mount survives a restart, and Proxmox checks it is mounted**

The creation workflow configures a UUID-based systemd mount and sets the storage definition's `is_mountpoint` property, so when the filesystem is unmounted Proxmox treats the storage as unavailable instead of writing, without any warning, into the empty directory on the system filesystem. Validate the mount after creation and after any restart.

Restrict the directory to its role. In the validated release, a directory created with **Add Storage** initially permits Container, Disk image, ISO image, Backup, Container template, and Snippets. Open **Datcenter > Storage**, select `backup-local`, choose **Edit**, and on the **General** tab set:

FIELD	VALUE
Content	Backup only
Nodes	The intended node, such as <code>pve01</code>
Enable	Enabled
Shared	Disabled

Preallocation and **Allow Snapshots as Volume-Chain** sit under **Advanced** rather than on the General tab. Both apply to VM disk storage, and a backup-only filesystem has no use for either. Leave them as they are: the **Create: Directory** workflow does not turn volume-chain snapshots on, and a directory storage cannot have that setting changed after it is created.

Leave **Backup Retention** unchanged until a representative backup has been measured and a retention policy calculated (step 09). Unchanged means `keep-all=1`, so every generation is kept until you set a policy.

06 Point the system-disk storages at the right content

In a standard installation the default `local` directory at `/var/lib/vz` sits on the Proxmox system filesystem unless separate storage is mounted there; `findmnt --target /var/lib/vz` from step 01 confirms which. With dedicated backup storage in place, remove **Backup** from `local` so Proxmox no longer places backup archives on the system filesystem.

- Check `/var/lib/vz/dump` for archives written before this change. Removing the **Backup** content type leaves those files in place and stops Proxmox listing them, so they cannot be selected for restore. Move anything still needed to `/mnt/pve/backup-local/dump` first, or keep **Backup** enabled on `local` until they have been migrated or have aged out.
- Open **Datacenter > Storage**, select `local`, choose **Edit**, and deselect **Backup**. Keep ISO image and Container template, keep the storage enabled, and keep it not shared.
- The change deletes nothing and resizes nothing; it only controls which content types Proxmox may place there from now on.
- Add **Import** if the node will hold OVA or OVF files for Proxmox's guest-import workflow. A default installation does not enable it. That content type holds import sources such as OVAs and VM disk-image files. Running VM disks still cannot be placed on `local` unless **Disk image** is also enabled.
- Retain `local-lvm` as it is: LVM-thin, content Disk image and Container, enabled, not shared. It stays useful for small utility workloads, temporary migration space, or recovery operations. For routine VM creation in this design, select `vmfast` explicitly as the disk destination.

07 Validate the finished layout

Open **Datacenter > Storage** and compare against the target state:

ID	TYPE	CONTENT	SHARED	ENABLED
backup-local	Directory	Backup	No	Yes
local	Directory	ISO image, Container template (Import optional)	No	Yes
local-lvm	LVM-Thin	Disk image, Container	No	Yes
vmfast	LVM-Thin	Disk image, Container	No	Yes

Select `vmfast` in the server tree and confirm its reported capacity is consistent with the source disk; select `backup-local` and confirm it is active with the expected filesystem capacity. Then verify from the node shell:

```
pvesm status
lsblk -e 7 -o NAME,PATH,SIZE,MODEL,SERIAL,TYPE,FSTYPE,MOUNTPOINTS
lvs -a -o lv_name,vg_name,lv_size,data_percent,metadata_percent,devices
findmnt --target /mnt/pve/backup-local
mountpoint /mnt/pve/backup-local
df -hT /mnt/pve/backup-local
```

- `vmfast` is active and reported as LVM-thin storage; `backup-local` is active and reported as directory storage.
- The backup filesystem is mounted at `/mnt/pve/backup-local` using ext4, and `mountpoint` confirms an active mount.
- The LVM-thin data and metadata percentages are healthy, and no target disk is unexpectedly mounted or assigned elsewhere.
- Restart the node and repeat the `pvesm status`, `findmnt`, and `mountpoint` checks. The backup target depends on one systemd mount unit, and a mount that does not return after a restart leaves the storage unavailable until someone notices.

08 Prove it end to end with a restore

A backup task that finishes successfully tells you the archive was written, and nothing else. Run the whole loop before declaring the storage ready:

- Upload an approved installation ISO to `local`, create a small non-production VM, and select `vmfast` explicitly for its disk. Install and start the guest.
- Back up the VM to `backup-local`, confirm the task finishes successfully, and record the resulting backup size for step 09.
- Restore the backup to `vmfast` under a new VM ID, and tick **Unique** in the restore dialog so the restored guest gets fresh MAC addresses instead of the source guest's. Before starting the restored guest, isolate its network interfaces or power off the source guest, so the two guests cannot conflict through duplicate IP addresses, hostnames, or application identities.
- Start the restored VM, verify the required data and services, then remove the test VM, the test restore, and the test backup in accordance with the change procedure once validation is complete.

09 Size retention from a measured backup

Backups written to directory storage are full archives per generation. Compression shrinks each archive, and Proxmox stores every generation in full; it does not deduplicate one generation against another the way Proxmox Backup Server (PBS) does. Size the retention policy from measurement:

- Create a representative backup and measure its actual size (step 08 already produced one).
- Project growth in written guest data and in the number of retained generations, and preserve operational free space on the backup filesystem.
- Account for the ext4 root reserve. The **Create: Directory** workflow runs `mkfs` with its defaults, which hold back 5% of the filesystem for the super-user, about 93 GiB on a 2 TB disk. Proxmox counts that space in the storage's Total and leaves it out of Available, so the two never reconcile; `vzdump` runs as root and can write into the gap anyway. On a filesystem dedicated to backups, `tune2fs -m 1 /dev/sdb1` returns most of it, and Available then matches what a backup can use.
- Create the recurring job under **Datacenter > Backup** with `backup-local` as its target storage, and set its retention in the same sitting: a job with no retention options inherits `keep-all=1` and keeps every generation.
- Calculate how many daily, weekly, and monthly restore points fit safely, then configure pruning on the job or on the storage's **Backup Retention** tab; a job-level setting overrides the storage's.
- Configure Proxmox notifications for failed backup jobs, and point monitoring at storage capacity so low space raises an alert before backups start failing.

The nominal sizes of thin-provisioned VM disks are the wrong sizing input on their own: actual consumption depends on written guest data, compression, data type, and generations retained. Several mostly empty virtual disks may compress to small archives, while encrypted, already-compressed, database, or media data compresses poorly.

OPS Monitoring and maintenance

- Watch both data and metadata consumption on every LVM-thin pool. `lvs -o lv_name,vg_name,data_percent,metadata_percent` reports both, and `lvs -o+seg_monitor vmfast/vmfast` confirms `dmeventd` is monitoring the pool. LVM's automatic extension cannot rescue this layout, because the pool uses the whole volume group and there are no free extents to extend into, so wire the two percentages into a check that reaches a Proxmox notification target and act well before either approaches full.
- Maintain free capacity for snapshots, backup staging, guest growth, and administrative recovery. The volume group's nominal allocation does not show how much of that is left; the pool's data and metadata percentages do.
- Thin provisioning lets the sum of configured virtual disks exceed the physical pool. Avoid intentional overcommitment unless monitoring, alerting, capacity forecasting, and emergency expansion or migration procedures are all in place.
- Monitor backup filesystem capacity and verify retention pruning runs as intended.

- Confirm `smartd` is running and that its mail to root reaches a person. Proxmox enables `smartd` by default, scans the disks every 30 minutes, and mails root when it finds a problem; **Datacenter > Notifications** turns that mail into a notification event and delivers it to a configured target. No disk in this design is mirrored, so this mail is the only routine warning the host gives about a failing disk. A drive can also fail with no warning at all; the backup and restore path covers that case.
- Review S.M.A.R.T. health and SSD wear indicators periodically, and document every disk replacement by model, serial number, device role, and date.
- Treat the archives at the classification of the most sensitive guest they hold. Each one is a whole guest disk in a single file, and Proxmox creates the backup directory and the archives with default permissions, so check them with `ls -l /mnt/pve/backup-local/dump`: on a stock node any account with a shell on the host can read them. Host access is what protects them.
- Protect the host with a correctly sized, monitored uninterruptible power supply (UPS) and test controlled shutdown behavior.
- Apply supported Proxmox and Debian security and maintenance updates, test representative restores on a defined schedule, and review storage roles whenever workloads, retention requirements, or recovery objectives change.

Keep a backup copy off this host. The `backup-local` disk speeds up recovery, and it sits in the same physical server as the workloads it protects. For client or production systems:

- Maintain at least one additional copy on separate hardware or in another approved location, and define and test recovery time and recovery point objectives (RTO and RPO: how long recovery may take, and how much recent data may be lost).
- Prefer a dedicated Proxmox Backup Server where appropriate, for deduplication, integrity verification, access control, and efficient retention, and never install the only PBS instance on the hypervisor it protects.
- Protect backup credentials separately from routine hypervisor administration, and consider offline, immutable, or logically isolated copies where the risk assessment requires them.
- Proxmox encrypts backups only when the target is a Proxmox Backup Server, so the archives on `backup-local` are written in the clear. Where encryption is required, either send those backups to a Proxmox Backup Server with client-side encryption, or place the backup filesystem on an encrypted block device. Protect the recovery keys separately from the host, since a key stored only on this server is lost with it.
- RAID, ZFS mirrors, replication, VM snapshots, and LVM-thin snapshots improve availability or convenience; independent backups are still required alongside them.

HELP Troubleshooting

• A target disk does not appear in a creation dialog

Confirm it shows **Usage: No** under **Disks > Disks** and that no partitions, filesystems, LVM volumes, ZFS labels, Ceph assignments, or mount points remain; select **Reload**. Review `lsblk`, `pvs`, `vgs`, `lvs`, `zpool status`, and `findmnt` as applicable, and do not force the disk back into use until you know what previously owned it.

• vmfast shows less than 2 TB

The difference is normal and has two causes. The interface reports binary units, so a disk sold as 2 TB is about 1.82 TiB before anything is written to it. LVM then reserves the pool's metadata volume and a spare copy of it, up to 16 GiB each, which took the validated device to a pool of about 1.79 TiB. Compare against the physical volume size that `pvs` reports rather than against the number on the label.

- **LVM-thin usage is unexpectedly high**

Review VM disk placement and actual guest consumption, and remove only confirmed obsolete snapshots and volumes. Confirm discard/TRIM (the guest telling the storage which blocks are free) is configured for the guest and storage path before relying on space reclamation. Check both **Data%** and **Meta%**, and migrate workloads or expand capacity before the pool fills.

- **The backup filesystem is filling rapidly**

Review backup sizes and retention settings, confirm pruning is enabled and completing, and check for ISO files, VM disks, templates, or manually copied files that have no place on `backup-local`. Reduce retention only in line with recovery requirements, and add separate backup capacity rather than aggregating disks into a failure-prone span.

- **A scheduled backup cannot select backup-local**

Confirm the storage is enabled for the correct node, its content type is **Backup**, and `/mnt/pve/backup-local` is mounted and writable. Review `pvesm status` and the task log for the specific error.

- **The wrong disk was selected**

Stop immediately. Create no new filesystems, LVM volumes, or workloads on the affected disk, because further writes reduce the chance of recovery. Preserve the current state and engage an approved data-recovery process if required.

DONE What you achieved

- Before every destructive step you identified the disk by device path, capacity, model, and serial number, recorded the disk-to-role mapping, and excluded the system disk.
- `vmfast` holds VM and container disks on the NVMe thin pool, locked to Disk image and Container content.
- `backup-local` is an ext4 directory locked to **Backup** content, mounted persistently by UUID, and reported unavailable when the filesystem is not mounted, so nothing writes into the empty directory underneath.
- The system disk keeps installation media and templates on `local` and no backups, with `local-lvm` retained as secondary capacity, so backup jobs no longer write to the system filesystem.
- The spare disk is initialized, documented, and deliberately unassigned.
- The layout passed a full backup-and-restore loop, and retention is sized from a measured archive. An off-host copy is still outstanding for anything critical; the operations section says what it takes.

REF References

Proxmox VE Administration Guide: Storage, Backup and Restore, and Host System Administration chapters · pve.proxmox.com/pve-docs

Proxmox VE Storage Manager, `pvesm(1)` · pve.proxmox.com/pve-docs/pvesm.1.html

Proxmox VE wiki: Storage: LVM Thin, and ZFS on Linux · pve.proxmox.com/wiki

Proxmox pve-storage 9.1.8, commit cd5c90c: Diskmanage.pm, API2/Disks.pm, API2/Disks/Directory.pm, API2/Disks/LVMThin.pm, Storage/DirPlugin.pm, and CLI/pvesm.pm · github.com/proxmox/pve-storage

Proxmox pve-manager and proxmox-widget-toolkit: storage/Base.js, window/Restore.js, panel/DiskList.js, and button/Button.js · github.com/proxmox

Linux manual pages: `lvmthin(7)` and `mke2fs(8)` · man7.org/linux/man-pages

OpenZFS documentation: `zpoolconcepts(7)` and `zpoolprops(7)` · openzfs.github.io/openzfs-docs

Proxmox Backup Server: Installation, and Technical Overview · pbs.proxmox.com/docs