anoBytes | TECHNOLOGY SOLUTIONS

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

 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 2026-08-16.

3 disk roles

fast VM pool · local backup target · documented spare

9 setup steps

back up a test VM, restore it, boot it, then size how many backups to keep

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. Every destructive step is gated by positive disk identification, and the finished layout is proven 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 design thinking still informs clustered environments, and the steps here assume one node.

The permissions the work needs

In the current Proxmox API, disk creation requires the `Sys.Modify` privilege on `/`, and enabling **Add Storage** also requires `Datastore.Allocate` on `/storage`. 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
`/dev/nvme0n1`	Intel NVMe SSD	512 GB	Existing Proxmox system disk	`local` and `local-lvm`
`/dev/nvme1n1`	Seagate FireCuda 520 NVMe SSD	2 TB	Primary VM and container storage	`vmfast` (LVM-thin)
`/dev/sdb`	Crucial MX500 SATA SSD	2 TB	Local backup target	`backup-local` (ext4 directory)
`/dev/sda`	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
local	Directory at `/var/lib/vz`	Import, ISO image, Container template	Import sources, installation media, and templates on the system disk
local-lvm	LVM-thin	Disk image, Container	Existing system-disk thin pool, retained as secondary local capacity
vmfast	LVM-thin	Disk image, Container	Primary VM and container disk storage
backup-local	Directory at `/mnt/pve/backup-local`	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. The trade-off is operational: a thin pool that fills up can interrupt guest writes and put data at risk, so watch both data and metadata usage on the pool, and do not overcommit capacity without monitoring and a documented response. Regular LVM remains the fit where fully reserved, highly predictable allocation is the primary requirement.

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 additional copy on separate hardware or in another approved location.

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, and 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, with the slower member shaping write performance, and it would consume the disk meant to be the backup target. A three-disk RAIDZ1 across 2 TB, 2 TB, and 1 TB is constrained by the smallest member, so it also lands near 2 TB usable while stranding capacity on the larger disks. A single-disk ZFS pool can detect corruption and offer snapshots, and without a second data copy it cannot repair ordinary corruption. An organization that puts continued operation through a disk failure ahead of a separate local backup target may reasonably choose a ZFS mirror instead; decide before creating LVM or directory storage, because changing the design later means migration and disk reconfiguration. ZFS snapshots and mirrors are still not backups.

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 actually does**
  In the Proxmox `pve-storage` implementation reviewed on 2026-08-16, **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 actually contains `/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 is unavailable or greyed out**
  The disk is likely still mounted, referenced by a Proxmox storage definition, assigned to LVM or ZFS, or otherwise in use. 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.

03 Initialize with GPT and pause at the checkpoint

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

This is a deliberate validation point: GPT exists, and 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.

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	`/dev/nvme1n1`, verified against the approved disk inventory
Name	`vmfast`
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 roughly the disk capacity after storage metadata and unit conversion. The validated 2 TB disk appeared as about 1.97 TB, which is 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 **Datacenter > 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	`/dev/sdb`, verified against the approved disk inventory
Filesystem	`ext4`
Name	`backup-local`
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 is persistent, and the storage knows it is a mount point**
  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 quietly writing 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 **Datacenter > Storage**, select **backup-local**, choose **Edit**, and on the **General** tab set:

FIELD	VALUE
Content	**Backup** only
Nodes	The intended node, such as `pve01`
Enable	Enabled
Shared	Disabled
Preallocation	Default
Allow Snapshots as Volume-Chain	Disabled

The volume-chain snapshot option applies to VM disk storage, and a backup-only filesystem has no use for it. Leave **Backup Retention** unchanged until a representative backup has been measured and a retention policy calculated (step 09).

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.

- Open **Datacenter > Storage**, select **local**, choose **Edit**, and set **Content** to Import, ISO image, and Container template, with **Backup** deselected. Keep the storage enabled and not shared.
- The change deletes nothing and resizes nothing; it only controls which content types Proxmox may place there from now on.
- The **Import** content type holds import sources such as OVAs and VM disk-image files for Proxmox's guest-import workflow. It does not turn `local` into a home for running VM disks 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.

- **The spare disk stays empty on purpose**
  The 1 TB `/dev/sda` stayed initialized with GPT and otherwise unused (**Usage: No**, **GPT: Yes**). An unassigned disk preserves flexibility: 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.

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	Import, ISO image, Container template	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,TYPE,FSTYPE,MOUNTPOINTS,MODEL,SERIAL
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.

08 Prove it end to end with a restore

A successful backup task proves the archive was written; the restore test proves it can be used. 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. 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 them, and separate generations are never deduplicated against one another the way Proxmox Backup Server (PBS) stores them. 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.
- Create the recurring job under **Datacenter > Backup** with `backup-local` as its target storage.

- 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

- Monitor both data and metadata consumption for every LVM-thin pool, with warning and critical thresholds set well before a pool approaches full.
- Maintain free capacity for snapshots, backup staging, guest growth, and administrative recovery, and review both figures rather than only the nominal volume-group allocation.
- 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.
- 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.
- 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.

Backup resilience beyond this host. The `backup-local` disk is a useful local recovery layer that remains inside the same physical server as the workloads. 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.
- When encryption is required, use a supported mechanism appropriate to the backup target and protect the recovery keys independently.
- 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 hold off on forcing reuse until the previous ownership is understood.

- **vmfast shows less than 2 TB**

  Disk manufacturers, operating systems, storage metadata, and user interfaces represent capacity differently, so a value slightly below the marketed size is normal. The validated 2 TB device produced an approximately 1.97 TB thin pool.

- **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

- Every destructive step was gated by positive identification: device path, capacity, model, serial number, and a recorded disk-to-role mapping, with the system disk explicitly excluded.
- `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 only Import, ISO image, and Container template content on `local`, 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, retention is sized from a measured archive, and an off-host backup copy is planned or operational for anything critical.

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 source, commit cd5c90c: Diskmanage.pm and API2/Disks/Directory.pm · github.com/proxmox/pve-storage

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

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