

N9LYA TECHNICAL GUIDES

Proxmox VE

Virtualization From the Ground Up

Jerome M. Kutche

N9LYA

Second Edition · September 2026

About This Book

Proxmox VE occupies an unusual position. It is simultaneously the easiest serious hypervisor to install — twenty minutes from ISO to a working web interface — and one of the least forgiving to operate badly, because it hands you the full Linux stack underneath and assumes you know what you’re doing with it.

That gap is where this book lives.

The installation guides are everywhere. What’s harder to find is the reasoning: which storage type to choose and what you’re committing to when you choose it, when a container is the right answer and when it will waste a weekend, why your cluster fenced a node at 3 a.m., what actually happens during a major version upgrade, and how to run the thing for three years without it slowly becoming a system you’re afraid to touch.

Why Now

Two things make this a particularly good moment to learn Proxmox properly.

The first is that a very large number of organizations and individuals are migrating to it from other platforms, driven by licensing changes elsewhere. Many arrive with VMware habits that map badly onto Proxmox and produce configurations that work but shouldn’t. Chapter 27 covers migration specifically.

The second is a version boundary. Proxmox VE 9 is built on Debian 13 “Trixie” with a substantially newer kernel, and the 8.x line reached the end of its support window in August 2026. If you’re on 8.x, you have an upgrade to plan — and it is a real upgrade with real breaking changes, not a routine update. Chapter 20 covers it properly.

What This Book Covers

Everything from a first installation through clustering, Ceph, GPU passthrough, and the operational discipline that keeps a system healthy. It assumes you know your way around Linux at a basic level — you can edit a file, read a log, and use SSH. It does not assume you know anything about virtualization.

Where there are several ways to do something and one is right for most people, I say which and why. Where the popular forum answer has a consequence nobody mentions, I say that too.

What It Isn’t

It is not a replacement for the official documentation, which is genuinely good and covers every parameter of every command. This book covers what the documentation doesn’t: which parameters matter, in what order to do things, and what happens when you get it wrong.

It is not a Linux tutorial, though it explains the Linux mechanisms that matter.

It is not a certification guide. Proxmox doesn’t have one.

Versions

Command syntax and behavior in this book reflect Proxmox VE 9.x. Most of it applies unchanged to 8.x, and I note where it doesn’t. Proxmox uses a rolling release model, so the specific version numbers here will age — check the official documentation for your installed version when something doesn’t match. Chapter 20 covers how to know what you’re running and how to move forward safely.

How to Use It

Each chapter ends with a **Console Check** — verification commands and short exercises. Run them. Reading a configuration and confirming a configuration are different activities.

Commands are given in the CLI, because the CLI is what you'll use when the web interface won't load, what you'll script, and what every forum answer will assume. The web interface maps onto it closely; learning one teaches you the other.

Anything capable of destroying data or disconnecting you is marked. Take those seriously — Proxmox will let you do almost anything, including things you'll regret.

About the Author

Jerome M. Kutche holds the amateur radio callsign N9LYA, first licensed in 1991. He spent over three decades in industrial electrical and controls work — Allen-Bradley and Siemens PLC systems, motor control, plant electrical maintenance and supervision across manufacturing and heavy industry — and holds NFPA 70E, lockout/tagout, and thirty-nine other professional certifications.

He serves as Indiana's 44Net/AMPRNet coordinator and is trustee for two amateur radio clubs. He runs Proxmox on enterprise hardware — an HP ProLiant DL380 with dual GPUs passed through for local AI inference, ZFS underneath, a dozen containers and virtual machines, and a backup server that has been tested by actual restores rather than assumed. Much of what's in this book was learned by getting it wrong first.

Part I — Foundations

Chapter 1: What Proxmox VE Actually Is

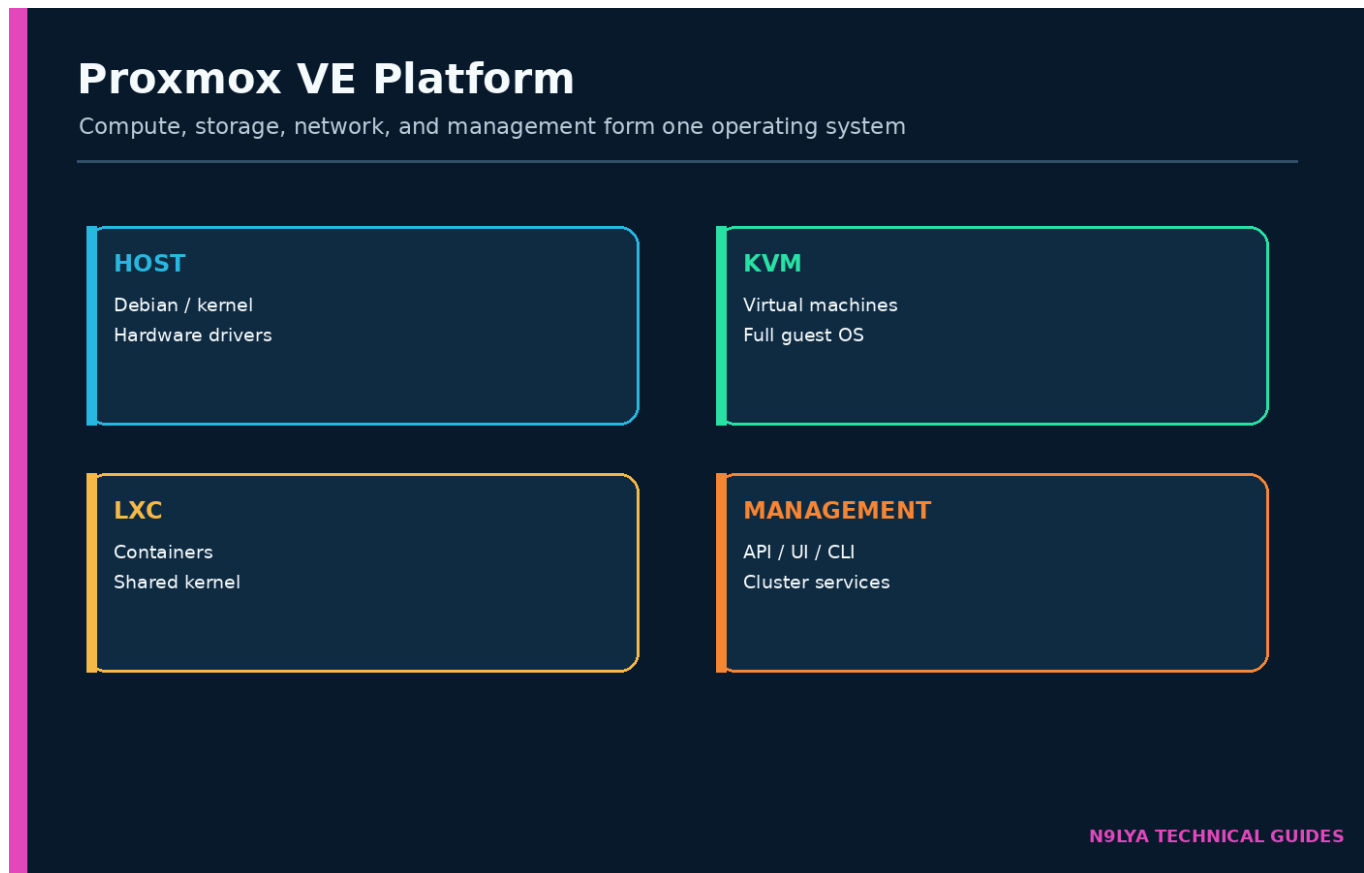


Figure 1: The Proxmox VE platform

Understanding what Proxmox is — and specifically what it is assembled from — explains most of its behavior, including the parts that surprise people.

The Stack

Proxmox VE is not a hypervisor. It is a management layer over several existing Linux technologies, integrated and given a web interface.

Debian GNU/Linux is the base operating system. Proxmox VE 9 sits on Debian 13 “Trixie”. This is the single most consequential fact about the platform: **your hypervisor is a Debian server**, and everything you know about Debian applies. You can install packages, edit configuration files, run cron jobs, and read logs the normal way.

It also means Debian’s problems are your problems. A misconfigured repository, a full root filesystem, a broken package upgrade — these affect your hypervisor exactly as they would any Debian machine.

KVM provides hardware virtualization. It is part of the Linux kernel, which is why Proxmox has no separate hypervisor layer to install or patch. **QEMU** provides the device emulation around it — virtual disks, network cards, chipsets.

LXC provides system containers, sharing the host kernel and isolating at the process level.

ZFS is available as a first-class storage option, integrated into the installer and the web interface. **Ceph** is available for distributed storage in clusters.

Corosync handles cluster communication and membership. **pmxcfs** — the Proxmox Cluster File System — is a database-backed filesystem mounted at `/etc/pve` that replicates configuration across all cluster nodes automatically.

The management layer itself is written in Perl and, increasingly, Rust, and it’s licensed under the AGPL.

What This Architecture Means

You have a full Linux system. This is Proxmox’s greatest strength and its most common source of self-inflicted damage. You can do anything. Including things that leave the management layer’s view of the world inconsistent with reality.

The rule that follows: make changes through Proxmox’s tools where Proxmox’s tools exist. Creating a VM disk manually with `qemu-img` produces a disk Proxmox doesn’t know about. Editing `/etc/network/interfaces` directly works, but the web interface stages network changes differently and you can end up fighting it. Use `qm`, `pct`, `pvesm`, and the interface; drop to raw Linux tools when there’s no Proxmox equivalent.

/etc/pve is special. It looks like a directory. It is a FUSE filesystem backed by a replicated database. Files written there appear on every cluster node within moments. It is also **read-only when the node loses quorum**, which is a behavior that confuses people badly the first time they encounter it — you cannot start a VM or edit a configuration on a node that has lost cluster quorum, because writing the configuration is impossible.

Nothing is hidden. Every VM configuration is a text file in `/etc/pve/qemu-server/`. Every container is in `/etc/pve/lxc/`. Storage definitions are in `/etc/pve/storage.cfg`. You can read them, back them up, diff them, and put them in version control.

Licensing and Subscriptions

Proxmox VE is free and open source. There is no feature-limited free tier, no node limit, no artificial restriction. Clustering, high availability, Ceph, backup — all included.

What a subscription buys is **access to the enterprise repository** and support. The enterprise repository receives packages after additional testing; the no-subscription repository gets them earlier. The software is identical.

For a homelab or a lab environment, the no-subscription repository is entirely appropriate. For anything you depend on, a subscription is inexpensive relative to the cost of a bad update, and it funds the project.

If you use the no-subscription repository, you will see a nag dialog at login. It is cosmetic. Chapter 4 covers configuring repositories correctly.

The Competition, Honestly

VMware ESXi / vSphere. The enterprise standard for two decades, and the reason many readers are here — licensing and packaging changes have made small deployments considerably less attractive. vSphere remains more polished in places and has a deeper ecosystem of third-party integrations. Proxmox matches or exceeds it on cost, on storage flexibility, and on not requiring a separate management appliance.

XCP-ng. Open source, mature, Xen-based, with strong management tooling in Xen Orchestra. A legitimate alternative. Smaller community, and the best management experience involves either a paid appliance or building from source.

Hyper-V. Sensible if your environment is Windows-centric and your team is comfortable there.

Plain KVM with libvirt. What Proxmox is built on, without the management layer. Maximum flexibility, minimum assistance.

Docker or Kubernetes alone. Not hypervisors. If every workload you have is containerizable, you may not need virtualization at all. The moment you need a Windows VM, a different kernel, or PCIe passthrough, you do.

Where Proxmox Genuinely Wins

Containers and VMs in one interface. LXC containers use a fraction of the memory of full VMs for workloads that don't need kernel isolation. Having both, managed identically, is a real advantage and one that most competitors don't offer.

ZFS integration. Snapshots, replication, checksummed integrity, and compression, configured from the installer.

Backup that's actually good. The integrated backup system plus Proxmox Backup Server — deduplicating, incremental, encrypted, with verification jobs that read stored data and confirm it's intact — is better than what many commercial platforms include.

No licensing games. Cluster three nodes without buying anything.

It's Debian. Anything you can run on Linux, you can run on the host if you must. This is a double-edged capability and Chapter 4 argues for restraint.

What's New Enough to Matter

Proxmox VE 9 moved to Debian 13 and a much newer kernel, which improves hardware support and changes some behaviors around network interface naming and LVM activation — both of which matter during upgrades.

The 9.2 release added dynamic load balancing, so a cluster can migrate guests automatically to balance load; the ability to disarm the high-availability stack cluster-wide during maintenance without risking node fencing; expanded software-defined networking; and management of custom CPU models from the web interface.

Proxmox also now offers builds for ARM64 on specific enterprise platforms, a first for the project. For the overwhelming majority of readers this is a footnote — assume x86-64 unless you specifically know otherwise.

Chapter 20 covers version strategy and how to keep current safely.

Console Check

1. Run `pveversion -v` and record every component version.
2. Look at `/etc/pve/qemu-server/`. Read a VM's configuration file.

3. Run `cat /etc/os-release` and confirm which Debian release you're on.
4. Run `pvesm status` and note the storage available.

Chapter 2: Planning the Build



Figure 2: Planning by failure domain

Time spent here saves rebuilds later. Several Proxmox decisions are difficult or impossible to change after installation.

The Decisions That Are Hard to Undo

Boot filesystem. Choosing ext4, XFS, or ZFS for the root filesystem at install time is effectively permanent. Changing it means reinstalling.

ZFS pool topology. A pool's vdev layout — mirrors versus RAIDZ, and how many disks per vdev — generally cannot be restructured after creation. You can add vdevs; you cannot convert a RAIDZ2 into mirrors.

Cluster membership. Adding a node to a cluster is straightforward. Removing one cleanly is fiddly, and a node that has ever been in a cluster should be reinstalled before joining a different one.

Node naming. Renaming a node in an existing cluster is painful enough that the practical answer is “don't.”

Everything else — storage additions, network changes, VM configuration — is adjustable.

Hardware Requirements

Minimum to run at all: a 64-bit x86 CPU with virtualization extensions, 2 GB of RAM, and a disk.

Minimum to be useful: 4 cores, 16 GB RAM, an SSD.

Comfortable: 8+ cores, 32-64 GB RAM, NVMe for VM storage plus bulk storage as needed.

The specifics that matter:

Virtualization extensions — Intel VT-x or AMD-V. Present on essentially everything modern, and **frequently disabled in BIOS by default**. This is the single most common “why won’t my VM start” answer. Verify it:

```
grep -E 'vmx|svm' /proc/cpuinfo | head -1
```

Empty output means it’s off in firmware or unsupported.

IOMMU — Intel VT-d or AMD-Vi — is required for PCIe passthrough. If you intend to pass through a GPU, a storage controller, or a network card, verify both CPU and motherboard support. Consumer motherboards are inconsistent about IOMMU group separation; server and workstation boards are much better. Chapter 24 covers this in detail.

ECC memory is worth having, particularly with ZFS, which assumes memory is trustworthy when it verifies checksums. Not mandatory. Used workstations and servers provide it cheaply.

RAM is the binding constraint. Every VM reserves memory whether it uses it or not. Containers are far lighter. Plan for your guests plus 2-4 GB for the host, plus whatever you allocate to ZFS caching.

Storage Planning

This deserves more thought than most people give it, because storage decisions constrain everything downstream.

Separate the boot device from VM storage where you can. Proxmox itself needs very little — 32 GB is plenty for the operating system — and mixing host and guest I/O on one device causes contention.

Use SSD or NVMe for VM disks. The difference between spinning disks and flash for virtual machine storage is the difference between a system that feels responsive and one that feels broken. If you buy exactly one fast device, make it the one your guests live on.

Spinning disks are fine for bulk storage — backups, media, archives, ISO images.

Two structural rules for ZFS:

Never run ZFS behind a hardware RAID controller. ZFS wants direct access to disks so it can detect and repair errors. A RAID controller hides the disks behind an abstraction that defeats this. Use an HBA in IT mode — LSI 9200 or 9300 series cards flashed to IT firmware are the standard, inexpensive answer.

Plan your vdev layout before you build, because you can’t restructure it afterward. A pool loses all data if any single vdev fails, so adding a vdev adds capacity and a failure domain simultaneously.

Never plan to run above 80% full. Copy-on-write filesystems fragment badly and slow dramatically when nearly full. ZFS degrades noticeably past 80% and severely past 90%.

Network Planning

One interface works. Two is much better, and here is why: with a second interface you can keep a management path that is independent of whatever you’re doing to the bridge. Misconfiguring the bridge on a headless single-interface host means a trip to the physical console.

More interfaces are more useful than faster ones for most homelab and small-office builds. Giga-bit is genuinely sufficient for most workloads. 10G matters for storage replication, Ceph, and moving large images.

Plan your addressing before installing, because the installer asks for a static address and changing a node’s address later — especially in a cluster — is disruptive.

Decide your VLAN scheme up front if you intend to segment. Chapter 14 covers implementation, but knowing which VLANs exist before you configure the bridge saves rework.

Sizing Guests

CPU can be overcommitted freely. Assigning 24 total vCPUs across guests on an 8-core host is normal, because most guests idle most of the time.

But **don't give a single VM many vCPUs without reason.** The scheduler may need to find that many free cores simultaneously. Start at 2 and increase when you observe a need.

Memory should not be overcommitted the way CPU is. When a host exhausts RAM it swaps, and performance collapses catastrophically rather than degrading gracefully. Assign what guests actually need, monitor real usage, and adjust downward — most VMs are given far more than they use.

Reserve for the host. 2–4 GB minimum, more with ZFS, which will consume free memory as cache and release it under pressure. That release is not instantaneous, which is why leaving genuine head-room matters.

A Sizing Example

A modest but real homelab: reverse proxy, DNS with filtering, monitoring, a file server, home automation, a documentation wiki, a Git server, and two VMs for experiments.

Guest	Type	RAM	vCPU
Reverse proxy	LXC	512 MB	1
DNS/filtering	LXC	512 MB	1
Monitoring	LXC	1 GB	2
File server	LXC	2 GB	2
Home automation	VM	2 GB	2
Wiki	LXC	1 GB	1
Git server	LXC	1 GB	2
Lab VM ×2	VM	4 GB	4
Guest total		12 GB	15 vCPU
Host + ZFS ARC		6 GB	—
System total		18 GB	8 cores fine

Note how much of that is containers. Running those seven services as full VMs would roughly double the memory requirement for no benefit.

Console Check

1. Verify virtualization extensions are present and enabled.
2. If you plan passthrough, verify IOMMU support in CPU and motherboard.
3. Decide your root filesystem and, if ZFS, your vdev layout. Write it down.
4. Total your planned guest memory, add host overhead, and compare to installed RAM.
5. Reserve static IP addresses for your nodes before installing.

Chapter 3: Installation

The installation is genuinely easy. The choices inside it are the part that matters.

Preparing Media

Download the ISO from the official site and **verify the checksum.** The download page publishes a SHA256 value; compute it and compare.

```
sha256sum proxmox-ve_9.2-1.iso
```

This takes ten seconds and it is the one verification step worth never skipping — a corrupted or tampered installer is a bad way to start.

Write it to USB with a tool that does a raw image write. On Linux, `dd` works. Some graphical tools reformat the media in ways that break the Proxmox installer specifically; if the installer won't boot, this is usually why.

Firmware Settings

Before installing, in the system firmware:

- **Enable virtualization extensions** (VT-x / AMD-V)
- **Enable IOMMU** (VT-d / AMD-Vi) if you'll ever want passthrough — enable it now even if you're unsure, as it costs nothing
- **Decide UEFI or legacy boot.** UEFI is the right choice on modern hardware, and it's required for some features. Be consistent — installing in one mode and booting in the other produces confusing failures.
- **Disable Secure Boot** initially unless you specifically need it. Proxmox supports it, and it adds complexity to passthrough and third-party kernel modules.
- **Set the correct boot order**, and consider disabling boot from devices you don't use.

The Installer

Modern Proxmox releases offer both a graphical and a terminal installer, plus an automated installation mode for unattended deployment. The graphical installer is fine for a first build.

The filesystem choice is the significant decision:

ext4 — simple, well understood, low overhead. Reasonable if your VM storage is elsewhere and you want the boot device to be uncomplicated.

XFS — similar, good with large files.

ZFS (RAID0/1/10/Z1/Z2/Z3) — the installer can create a ZFS root pool across multiple disks. This gives you boot redundancy, snapshots of the host itself, and the ability to roll back a bad upgrade. It costs memory and adds complexity.

The recommendation for most builds: ZFS mirror on two devices for the boot pool if you have two devices to spare, ext4 on a single device if you don't. Then a separate ZFS pool for VM storage on your faster or larger disks.

Advanced options in the installer let you set the ZFS ashift value, compression, and how much of each disk to use. Setting `hdsize` smaller than the full disk leaves free space you can use later — useful on a large boot device.

Network configuration asks for hostname (as a fully qualified domain name), IP address, gateway, and DNS server. Use a real FQDN — `pve1.lab.example.com`, not `pve1` — because certificate generation and cluster operations use it.

The address you set here becomes the management address and the address the cluster uses. Changing it later is disruptive. Get it right.

First Boot

The system boots to a console showing the management URL:

```
https://your.ip.address:8006
```

You will get a certificate warning. That's expected — the installer generates a self-signed certificate. Chapter 4 covers fixing it.

Log in with `root` and the password you set, realm "Linux PAM standard authentication".

Installing on Existing Debian

You can install Proxmox on top of an existing Debian system, which is useful when a hosting provider only offers Debian images, or when you need a specific partitioning scheme the installer won't produce.

The process adds the Proxmox repository, installs the Proxmox kernel and packages, and reboots into the Proxmox kernel. The official documentation covers it precisely and should be followed exactly, matching the Debian version to the Proxmox version — Debian 13 for Proxmox VE 9.

For most people, the ISO installer is better. Use the Debian path only when you have a reason.

Verifying the Install

```
pveversion -v      # component versions
pvesm status       # storage
ip a               # network
lsblk              # disks
zpool status       # if using ZFS
systemctl --failed # anything broken
```

Then check the basics: can you reach the web interface, does the node show as online, is there free space where you expect it.

Console Check

1. Verify the ISO checksum before writing media.
2. Confirm virtualization and IOMMU are enabled in firmware.
3. Install, choosing your filesystem deliberately.
4. Verify with the commands above.
5. Note your node name and IP — you'll need both if you cluster later.

Chapter 4: The First Twenty Minutes

A fresh Proxmox install is functional and not yet configured properly. These are the things to do immediately, in order.

Repositories

Out of the box, the enterprise repository is enabled and will fail to update without a subscription. Configure this first or every subsequent step produces errors.

Proxmox VE 9 uses the Debian deb822-style repository format. The files live in `/etc/apt/sources.list.d/`.

If you have a subscription: enter the key under **Datacenter → Subscription**, and leave the enterprise repositories enabled.

If you don't: disable the enterprise repositories and enable the no-subscription repositories. This is doable from the web interface under **Node → Updates → Repositories**, which is the safest way — it edits the right files in the right format.

If you also run Ceph, its repository needs the same treatment.

Do not enable the test repository on anything you care about.

Then update:

```
apt update && apt full-upgrade
reboot
```

Use `full-upgrade` (equivalently `dist-upgrade`) rather than `plain upgrade` — Proxmox package transitions sometimes require removing or installing packages, which `plain upgrade` will not do, leaving you partially updated.

Removing the Subscription Notice

The no-subscription repository produces a dialog at every login. It is cosmetic and it is also a reminder that you're on the earlier-testing repository, which is arguably useful information.

Patches to remove it exist and circulate widely. Be aware that they modify a packaged JavaScript file, which means they are undone by updates and must be reapplied. If it bothers you, the honest fix is a subscription — it's inexpensive and it funds the project you're relying on.

Time

```
timedatectl set-timezone America/Indiana/Indianapolis  
timedatectl status
```

Verify NTP synchronization is active. Clock drift between cluster nodes causes real problems, and logs with wrong timestamps make every future investigation harder.

A Real Certificate

The self-signed certificate produces a browser warning on every visit, which trains you to click through security warnings — a bad habit to cultivate.

Proxmox has built-in ACME support. Under **Node → Certificates → ACME**, you can register an account and request a certificate.

The DNS-01 challenge is what you want for an internal host, because it doesn't require your management interface to be reachable from the internet. Proxmox supports a wide range of DNS providers for this. You need a domain you own and API access to its DNS.

The result: a publicly trusted certificate on an internal-only management interface, renewed automatically, with no browser warnings and no root certificate to install on every device.

Email Notification

Proxmox sends notifications about backup results, ZFS pool problems, failed jobs, and system events. Out of the box these go to root's local mailbox, which nobody reads.

Configure it under **Datacenter → Notifications**. Recent Proxmox versions support several notification targets beyond SMTP, including webhook-based targets and Gotify.

Then test it. Send a test notification and confirm it arrives. A notification system that doesn't notify is worse than none, because you believe you're covered.

This matters more than it sounds. Proxmox will tell you about a degraded ZFS pool, a failed backup, and a disk throwing errors — but only if the message reaches a human.

Users

Do not use root for daily work.

```
pveum user add jerry@pam  
pveum acl modify / --users jerry@pam --roles Administrator
```

Then set a password for the corresponding Linux user, or configure another authentication realm. Chapter 18 covers permissions properly.

Set up two-factor authentication for administrative accounts under **Datacenter → Permissions → Two Factor**. Proxmox supports TOTP and WebAuthn hardware keys. A hypervisor management interface is one of the most valuable targets in your environment; treat it accordingly.

SSH Hardening

```
ssh-copy-id root@your.node
```

Then in `/etc/ssh/sshd_config`:

```
PasswordAuthentication no  
PermitRootLogin prohibit-password
```

Restart SSH and **verify key access works before closing your existing session**. This is the standard way people lock themselves out of a headless server.

Note that cluster nodes use SSH between themselves; don't restrict it in ways that break inter-node communication.

Network Access Restriction

The management interface should not be reachable from your general network, and certainly not from the internet.

Chapter 15 covers the Proxmox firewall. At minimum, decide now which subnet is your management network and plan to restrict port 8006 and SSH to it.

Never expose the Proxmox web interface to the internet. If you need remote access, use a VPN. Hypervisor management interfaces are actively scanned for and their compromise grants complete control of every guest.

Storage

The installer creates a default local storage for ISO images, container templates, and backups, and — depending on your filesystem choice — a local-lvm or local-zfs for guest disks.

Review under **Datacenter → Storage** and set the content types deliberately. Chapter 6 covers this properly, but the immediate task is knowing what you have and where guest disks will land.

Upload an ISO so you can build something: **Node → local → ISO Images → Upload**, or download directly from a URL, or place files in `/var/lib/vz/template/iso/`.

Download a container template: **Node → local → CT Templates → Templates**, which presents the available list.

The Restraint Argument

Proxmox is Debian, so you can install anything on the host. Docker, a media server, a database, a desktop environment.

Don't.

The host is the foundation everything else depends on. Software installed there can conflict with Proxmox packages, complicate upgrades, consume resources your guests need, and expand the attack surface of the one machine whose compromise means compromise of everything.

Run workloads in guests. That's what the guests are for. The exceptions are narrow: monitoring agents, backup agents, and hardware management tools.

Console Check

1. Configure repositories correctly and complete a full upgrade.
2. Set the timezone and verify NTP synchronization.
3. Obtain a real certificate via ACME with a DNS challenge.
4. Configure notifications and **send a test message**.
5. Create a non-root administrative user with two-factor authentication.
6. Verify key-based SSH works, then disable password authentication.

End of the free sample

Proxmox VE — Second Edition · by Jerome M. Kutche

That was the first 12 of 71 pages. The full book continues with the practical detail, worked examples and recovery steps that don't fit in a sample.

Get the full book (\$8.99, PDF + EPUB):

<https://booksrus.biz/product/proxmox-ve/>

Also on Gumroad:

<https://n9lya4.gumroad.com/l/drecmc>



All 17 Second Edition guides in one bundle for \$89.99:

booksrus.biz/product/n9lya-technical-guides-complete-bundle

This sample may be shared freely in unmodified form. © Jerome M. Kutche. Second Edition.