Homelab Starter Kit

Run local AI on a mini PC: from bare hardware to **your own private chat assistant** in 7 days. No per-token bill, no open ports on your router.



WHAT’S INSIDE PAGE

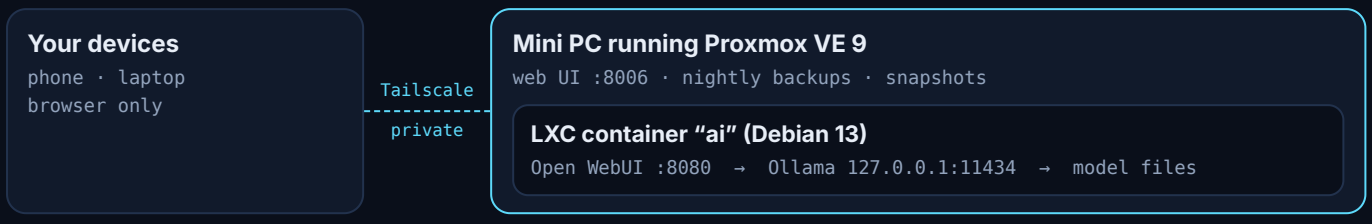
01 What you can realistically run 02
02 Hardware tiers 03
03 Model size vs RAM/VRAM 04
04 Install Proxmox VE 05
05 Your first LXC container 06
06 Ollama & your first model 07
07 Open WebUI 08
08 Remote access & security 09
09 Backups that restore 10
10 Monitoring & fixes 11
11 7-day starter plan 12
12 Glossary · Sources 13–14

HOW TO READ THIS KIT

Grey boxes are copy-paste commands; lines starting with `#` are comments. `<angle brackets>` are placeholders you replace. `[n]` points to a source on the last page. Anything tagged RULE OF THUMB is an approximation, not a benchmark.

01 What you can realistically run locally

Local AI is real and useful, but it is **not a frontier cloud model in a box**. Set expectations first, then buy hardware. This is the setup the kit builds:



TASK	REALISTIC LOCALLY?	START WITH
Chat, drafting, rewriting, summarising short texts	Yes, even CPU-only	3–4B models; 8–14B when memory allows (p4)
Coding help: explain code, small functions, shell one-liners	Yes; quality grows with model size	The largest model that fits fully in RAM or VRAM
Questions about your own files	Yes, through Open WebUI's document features	Your chat model; begin with short documents
Describe images and screenshots	Yes, with vision-capable models	e.g. `gemma4` accepts images [15]
Whole books or big codebases at once	Limited: context costs memory	Default context is 4k tokens below 24 GiB VRAM [10]
Frontier-level reasoning, hours-long agent runs	Not on this hardware	A cloud model; keep private data local
Image or video generation	Out of scope for this kit	Needs a GPU and a different software stack

THE HONEST TRADE-OFF

- **You win:** privacy, offline use, no per-token cost, full control.
- **You pay with:** speed, top-end quality, and your own time for updates and backups.
- On CPU, speed is limited mostly by **memory bandwidth**, not core count [23] (p4).

MEASURE, DON'T GUESS

This kit contains **no speed benchmarks**. Your RAM, cooling and model choice decide. Measure your own box:

```
$ ollama run --verbose llama3.2
# after each answer it prints timings,
# including "eval rate" in tokens/s
```

WHO THIS IS FOR

- You can copy commands into a terminal and read an error message.
- You have a machine you are allowed to wipe.
- You want to own the stack, not rent it – and you accept that you are the admin.

HAVE READY BEFORE DAY 1

- The mini PC (p3) and an Ethernet cable to your router
- A USB stick of 4 GB or more (the ISO is 1.71 GB [1])
- Monitor and keyboard for the install; a second computer with a browser
- A password manager for the new passwords

Rule for upgrades: start small, move up one model size at a time, and stop when answers stop feeling pleasant.

02 Hardware tiers

Three ways in, one software stack. **No prices on purpose:** RAM and GPU prices move fast, so compare current listings yourself. What fits where is a RULE OF THUMB based on p4.

TIER 1 Mini PC, CPU only

lowest power · quiet · simplest

LOOKS LIKE

A recent 6–8+ core mini PC, 32 GB RAM as **two matching sticks** (dual channel), 1 TB NVMe, wired Ethernet. 64 GB if the board supports it.

RUNS WELL

1–4B models comfortably, 7–8B usable, 12–14B slow. Mixture-of-experts models like `gemma4:26b` ("26B A4B"[15]) read fewer weights per token, so they run faster than dense models of similar size – but still need RAM for all weights.

WATCH OUT

Intel N100-class boxes have 4 cores, **one memory channel** and a 16 GB official maximum[22]: great for small services, weak for LLMs. A single RAM stick halves bandwidth on dual-channel boards.

TIER 2 Mini PC + eGPU, or small desktop + low-profile GPU

big speed-up for models that fit in VRAM

LOOKS LIKE

A mini PC with OCuLink or USB4/Thunderbolt plus an external GPU dock with its own power supply – or a small-form-factor desktop with a low-profile, slot-powered card (75 W class).

RUNS WELL

12–16 GB VRAM holds 8–14B models entirely on the GPU, e.g. `qwen3:14b` is a 9.3 GB download[14]. Leave headroom for context (p4).

WATCH OUT

Ollama supports NVIDIA (compute capability 5.0+, driver 550+), AMD via ROCm v7, and a Vulkan backend[11]. GPU passthrough on Proxmox is its own project and is not covered step by step here.

TIER 3 Used workstation tower

most headroom · most power & noise

LOOKS LIKE

A used business workstation with full-length PCIe x16 slots, a strong power supply and room for a 24 GB GPU (or two).

RUNS WELL

24 GB VRAM fits 27–32B models at 4-bit with modest context: `gemma3:27b` is 17 GB, `qwen3:32b` 20 GB[13][14].

WATCH OUT

Idle and load power, GPU power connectors (check before you buy the card), size and fan noise. Measure with a plug-in power meter.

BUYING CHECKLIST

- [ ] CPU virtualization (Intel VT-x / AMD-V) – required by Proxmox VE[4]
- [ ] RAM: two matching sticks; check the board's maximum
- [ ] NVMe SSD, 1 TB or more – models are 1–20 GB each[13][14]
- [ ] Wired Ethernet (2.5 GbE is nice to have)
- [ ] For a GPU: VRAM first, everything else second
- [ ] A second disk, NAS or USB drive for backups (p10)

Already own an Apple-silicon Mac or a gaming PC? Ollama runs there natively (Metal on Apple[11]). Use this kit for the server habits: containers, backups, remote access.

03 Choosing model size vs RAM/VRAM

The one question that matters: **does the model fit in fast memory, with room to spare for context?** Five rules, each sourced or labelled.

1 · SIZE ON DISK — RULE OF THUMB

≈ **0.6 GB per billion parameters** at Ollama's usual 4-bit default. Derived from library sizes: `llama3.2` 3B = 2.0 GB, `qwen3:8b` = 5.2 GB, `qwen3:14b` = 9.3 GB, `qwen3:32b` = 20 GB [13][14].

2 · RAM FLOOR — SOURCE

Ollama's README: at least **8 GB of RAM for 7B** models, **16 GB for 13B** and **32 GB for 33B** [13]. In a container, count only the RAM you gave it, minus what the OS and Open WebUI use.

3 · CONTEXT COSTS MEMORY — SOURCE

KV cache per token = 2 × layers × (heads × head size) × bytes. NVIDIA's example: Llama 2 7B at 4,096 tokens ≈ **2 GB** in 16-bit [23]. Many newer models need less, but long context is never free. Ollama defaults to 4k context below 24 GiB VRAM [10].

4 · GPU: FIT IT 100% — SOURCE

`ollama ps` shows `100% GPU`, `100% CPU` or a split such as `48%/52% CPU/GPU` [9]. Ollama's docs advise avoiding CPU offloading [10]: a split model runs much slower than one that fits in VRAM.

5 · CPU SPEED CEILING ≈ MEMORY BANDWIDTH ÷ MODEL SIZE — RULE OF THUMB

Generating text is memory-bound [23]: for a dense model, every new token reads roughly all weights once. Theoretical arithmetic, not a benchmark:

```
dual-channel DDR5-5600:  5600 MT/s × 8 bytes × 2 channels = 89.6 GB/s
5.2 GB model (qwen3:8b):  89.6 ÷ 5.2 ≈ 17 tokens/s  # ceiling; real numbers are lower
single channel DDR5-4800: 4800 MT/s × 8 bytes × 1 channel  = 38.4 GB/s  # N100-class
```

Model ladder — SIZES FROM OLLAMA.COM, OCT 2026

CLASS	EXAMPLES · DOWNLOAD SIZE	COMFORTABLE ON (rule of thumb)
Tiny 1–2B	`gemma3:1b` 815 MB · `qwen3:1.7b` 1.4 GB	Any box, even 8 GB RAM
Small 3–4B	`llama3.2` 2.0 GB · `qwen3:4b` 2.5 GB · `gemma3:4b` 3.3 GB	16 GB RAM, CPU only
Medium 8–14B	`qwen3:8b` 5.2 GB · `gemma3:12b` 8.1 GB · `qwen3:14b` 9.3 GB	32 GB RAM, or 12–16 GB VRAM
Large 27–32B	`gemma3:27b` 17 GB · `qwen3:32b` 20 GB	24 GB VRAM (CPU only: slow)

Sizes: Ollama README and library pages [13][14]. New families appear constantly, e.g. `gemma4` with tags `e2b`, `e4b`, `26b`, `31b` [15]: check the size on the model's library page, then apply rules 1–5.

WORKED EXAMPLE: 32 GB MINI PC, CPU ONLY, DUAL-CHANNEL DDR5-5600 — RULE OF THUMB

`qwen3:8b` · 5.2 GB
Fits easily in the ~28 GB container (p6). Ceiling ≈ 89.6 ÷ 5.2 ≈ 17 tokens/s. A good everyday start.

`qwen3:14b` · 9.3 GB
Fits with room for context. Ceiling ≈ 89.6 ÷ 9.3 ≈ 9.6 tokens/s: usable, noticeably slower.

`qwen3:32b` · 20 GB
Fits in RAM, but ceiling ≈ 89.6 ÷ 20 ≈ 4.5 tokens/s: sluggish for chat. A GPU tier job.

Ceilings are theoretical (rule 5); real speed is lower. Measure with `ollama run --verbose`.

04 Install Proxmox VE

Proxmox VE is a free, open-source hypervisor. Version 9 is based on **Debian 13 "trixie"**[4]; the current ISO at the time of writing is **9.2-1**[1].

1 Download and verify the ISO

Get the ISO from `proxmox.com/downloads`[1] and compare its checksum with the SHA256 on that page:

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

2 Write the USB stick (erases the stick)

```
# Linux, as root: find the stick (run before and after plugging it in), then write
lsblk
dd bs=1M conv=fdatasync if=./proxmox-ve_*.iso of=/dev/XYZ
```

Replace `/dev/XYZ` with the stick, never a disk you need. On Windows or macOS: Etcher works out of the box; Rufus works in **DD mode**[2].

3 Firmware (BIOS/UEFI) settings

Enable **Intel VT-x / AMD-V** (required[4]), plus VT-d / AMD IOMMU if you may add a GPU later. Boot from USB. If offered, set "power on after AC loss".

4 Installer choices

SCREEN	CHOOSE	WHY
Target disk → Options	**ext4** (the default)	Simplest. ZFS wants extra RAM: 4 GB + 1 GB per TB of raw disk[2]
Location	Country, time zone, keyboard	Correct logs and backup schedules
Password, email	12+ characters (8 is the minimum)[2]; an email you read	root controls everything; email receives notifications
Management network	Hostname such as `pve.home.arpa`; a fixed IP outside your router's DHCP range; gateway; DNS	The web UI lives at this address

5 First login

Browse to `https://<your-ip>:8006`, accept the self-signed certificate warning, and log in as `root` with realm **Linux PAM**[2].

6 Switch to the free update repository

The default enterprise repository needs a paid subscription. In the web UI: **your node → Updates → Repositories** → *Add* → "No-Subscription", then select each enterprise entry → *Disable*. Then **Updates → Refresh → Upgrade**, or in the host shell:

```
apt update && apt dist-upgrade
```

Prefer files? This is the official PVE 9 no-subscription entry[3]:

```
# /etc/apt/sources.list.d/proxmox.sources
Types: deb
URIs: http://download.proxmox.com/debian/pve
Suites: trixie
Components: pve-no-subscription
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

05 Your first LXC container

An LXC container shares the host's kernel: little overhead, starts in seconds, ideal for a CPU-only AI box. You will create **one container called `ai`** for Ollama and Open WebUI.

1 Download a Debian 13 template (host shell)

```
pveam update
pveam available --section system | grep debian-13
pveam download local <debian-13-standard_..._amd64.tar.zst>   # exact name from the list
pveam list local
```

Commands: pveam manual[6]. Same thing in the UI: **local storage → CT Templates → Templates**.

2 Create the container

Web UI: **Create CT** (top right). Use these values:

TAB	SETTING	VALUE
General	CT ID · Hostname	`101` · `ai`
General	Unprivileged · Nesting	both on (unprivileged is the default for new containers[5])
Template	Template	the `debian-13-standard` file you downloaded
Disks	Root disk	`local-lvm`, 64 GB – models are big
CPU · Memory	Cores · RAM · Swap	all cores · host RAM minus about 4 GB RULE OF THUMB · 512 MB
Network	IPv4	DHCP, then reserve the address in your router (or set a static IP)
Afterwards	Options → Features	add `keyctl` next to `nesting`: Docker and Tailscale inside use them[20]

Or the same in one command[5] (example values for an 8-core, 32 GB box):

```
pct create 101 local:vztmpl/<debian-13-standard_..._amd64.tar.zst> \
  --hostname ai --cores 8 --memory 28672 --swap 512 \
  --rootfs local-lvm:64 \
  --net0 name=eth0,bridge=vmbr0,ip=dhcp \
  --unprivileged 1 --features nesting=1,keyctl=1 --onboot 1
pct start 101
pct enter 101          # root shell inside the container, no password needed
```

3 First commands inside the container

```
apt update && apt full-upgrade -y
apt install -y curl openssl
passwd                 # optional: root password for console logins
```

HABIT: SNAPSHOT BEFORE EXPERIMENTS

```
# on the host
pct snapshot 101 clean-install
pct rollback 101 clean-install     # if an experiment goes wrong
```

A snapshot lives on the same disk as the container. It is an undo button, **not a backup** (p10).

06 Ollama and your first model

Ollama downloads, runs and serves models through one command-line tool and a local API on port 11434. Everything on this page runs **inside container 101**.

1 Install

```
curl -fsSL https://ollama.com/install.sh | sh
ollama -v
```

Official Linux method[8]; it also sets up a systemd service. Without a GPU the installer prints *"No NVIDIA/AMD GPU detected. Ollama will run in CPU-only mode."* – expected on Tier 1.

2 Run your first model

```
ollama run llama3.2               # 3B model, 2.0 GB download [13]
>>> Explain what a homelab is in two sentences.
>>> /bye
```

3 Everyday commands

COMMAND	WHAT IT DOES
`ollama list`	Models on disk (stored under `/usr/share/ollama/.ollama/models`[9])
`ollama ps`	Loaded models, CPU/GPU split, context size[9][10]
`ollama pull qwen3:4b`	Download a model without starting a chat
`ollama stop llama3.2`	Unload now; by default models unload after 5 minutes idle[9]
`ollama rm <model>`	Delete a model and free the disk space
`journalctl -e -u ollama`	Service logs[8]
`ollama run --help`	All flags for your installed version

4 Test the API (what Open WebUI talks to)

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [{"role": "user", "content": "Why is the sky blue?"}],
  "stream": false
}'
```

Base URL of the local API: `http://localhost:11434/api`[12].

KEEP THE API PRIVATE

Ollama listens on `127.0.0.1:11434` by default[9], so only programs inside this container can reach it. **Leave `OLLAMA_HOST` alone.** Open WebUI (next page) is the only door, and it has logins.

OPTIONAL: LOCAL-ONLY MODE

Ollama also offers cloud models. To switch its cloud features off[9], run `systemctl edit ollama.service` and add:

```
[Service]
Environment="OLLAMA_NO_CLOUD=1"
```

Then:
`systemctl daemon-reload && systemctl restart ollama`

Update later by re-running the install command[8].

07 Open WebUI

A browser chat interface with user accounts on top of Ollama. Still **inside container 101**.

1 Install Docker from Debian's own repository

```
apt install -y docker.io
docker version
```

Debian 13 ships Docker 26.1 as the package `docker.io`[18] – no third-party script needed.

2 Create a persistent secret key

```
openssl rand -hex 32 > /root/webui-secret && chmod 600 /root/webui-secret
```

Open WebUI's docs: set `WEBUI_SECRET_KEY` once and keep it, or every recreated container logs everyone out[16].

3 Start Open WebUI

```
docker run -d --network=host \
  -v open-webui:/app/backend/data \
  -e OLLAMA_BASE_URL=http://127.0.0.1:11434 \
  -e WEBUI_SECRET_KEY="$(cat /root/webui-secret)" \
  --name open-webui --restart always \
  ghcr.io/open-webui/open-webui:main
```

This is Open WebUI's documented host-network variant: the container reaches Ollama on `127.0.0.1`, and the UI listens on **port 8080** instead of 3000[17]. The first download is large; give it a few minutes.

4 First login

Find the container's address with `hostname -I`, open `http://<ct-ip>:8080` and choose **Create Admin Account**. The first account is the administrator, and new sign-ups are switched off once it exists[16]. Pick `llama3.2` in the model selector and chat.

5 Update

```
docker rm -f open-webui
docker pull ghcr.io/open-webui/open-webui:main
# then run the docker run command from step 3 again
```

Steps from Open WebUI's update guide[16]. Chats, users and settings live in the `open-webui` volume, so they survive the update.

DOCKER INSIDE LXC: HONEST NOTE

Proxmox recommends a QEMU VM for application containers when you need orchestration or live migration[4]. On one home server, Docker in an unprivileged container with `nesting` and `keyctl` is a common, workable setup – but not Proxmox's recommended path. **Snapshot before updates** (p6).

RATHER SKIP DOCKER?

Open WebUI also installs with `pip` or `uv`, but supports only Python 3.11 and 3.12; 3.13 is not supported yet[16]. Debian 13 ships Python 3.13, so you would need a separate Python version. Docker is the simpler path here.

08 Remote access and security basics

Rule zero: open no ports on your router. Nothing in this kit needs to be reachable from the internet. For access on the go, use Tailscale: a private network between your own devices.

1 Give the container a TUN device (host shell)

```
pct set 101 --dev0 /dev/net/tun
pct shutdown 101 && pct start 101
```

From Tailscale's LXC guide[20]. In the UI: container → **Resources → Add → Device Passthrough** → `/dev/net/tun`.

2 Install Tailscale and log in (inside container 101)

```
curl -fsSL https://tailscale.com/install.sh | sh
tailscale up            # open the printed link and sign in
tailscale ip -4         # the container's tailnet address, 100.x.y.z
```

Install command from Tailscale's Linux docs[19]. Then install the Tailscale app on your phone and laptop with the same account and open `http://ai:8080` (MagicDNS name) or `http://100.x.y.z:8080`.

3 Optional: HTTPS inside your tailnet

```
tailscale serve --bg 8080     # https://ai.<your-tailnet>.ts.net
tailscale serve status
tailscale serve reset         # undo
```

Serve shares a service **only within your tailnet**. Do not use *Funnel* for this: Funnel makes a service public on the internet[21].

SECURITY CHECKLIST

- ☐ **Router:** zero port forwards – not for 8006, 8080, 11434 or 22.
- ☐ **Ollama** stays on `127.0.0.1:11434` (default[9]); only Open WebUI faces the network.
- ☐ **Open WebUI:** one admin; keep sign-ups off, or approve "pending" users by hand[16].
- ☐ **Proxmox root:** long password plus two-factor login (Datacenter → Permissions → Two Factor).
- ☐ **Containers unprivileged** – the default for new ones[5].
- ☐ **Patch weekly:** host `apt update && apt dist-upgrade`; container `apt update && apt full-upgrade`; Ollama and Open WebUI as on p7–8.
- ☐ **Snapshot** before big changes; real backups per p10.
- ☐ **Privacy:** `OLLAMA_NO_CLOUD=1` keeps Ollama local-only (p7)[9].

TAILSCALE HABIT FOR AN ALWAYS-ON SERVER

Device keys expire after **180 days** by default, and connections to that device then stop[25]. For a trusted server, open the admin console's **Machines** page, find `ai`, open the menu on the right and choose **Disable Key Expiry**[25]. Keep expiry on for phones and laptops.

09 Backups that actually restore

A backup you have never restored is a hope, not a backup. Use **3-2-1**: three copies of your data, on two different types of media, one kept off-site [24].

COPY 1	COPY 2 · OTHER MEDIA	COPY 3 · OFF-SITE
The live container on the mini PC's NVMe.	Nightly Proxmox backup to a second disk, a NAS or a USB drive.	A rotated USB disk kept elsewhere, or an encrypted remote target such as a second Proxmox Backup Server.

What matters most: the `open-webui` volume (chats, users, settings) and your configs. Models can be downloaded again.

1 Add backup storage

Datacenter → Storage → Add: Directory (second disk), NFS, SMB/CIFS or Proxmox Backup Server. Not the system disk.

2 Schedule a nightly job: Datacenter → Backup → Add

FIELD	VALUE
Storage · Schedule	your backup storage · daily at 03:00
Selection	container 101 (and anything else you add later)
Mode · Compression	**Snapshot** (uses the storage's snapshot feature, minimal downtime) · **ZSTD** [7]
Retention	keep-daily 7, keep-weekly 4, keep-monthly 3 [7] – a starting point, adjust to your disk

One-off backup from the host shell [7]:

```
vzdump 101 --storage <backup-storage> --mode snapshot --compress zstd
```

3 Restore test (monthly, about 10 minutes, host shell)

```
# restore the newest backup as a NEW container 901 with a new MAC address
pct restore 901 <path-to>/vzdump-lxc-101-<date>.tar.zst --storage local-lvm --unique 1
pct set 901 --delete net0         # no network: avoids IP and Tailscale clashes
pct start 901
pct exec 901 -- ollama list       # are your models there?
pct exec 901 -- docker ps         # is open-webui running?
pct stop 901 && pct destroy 901   # clean up
```

Options from the `pct` manual [5]. Backup files sit in `/var/lib/vz/dump/` for the `local` storage, or in `dump/` under the mount point of other storages (e.g. `/mnt/pve/<storage>/dump/`). The UI alternative: storage → **Backups → Restore** with a new CT ID and "Unique" ticked.

DONE WHEN a restored copy booted and showed your models and Open WebUI. Write the date down; repeat monthly and after every big change.

10 Basic monitoring and quick fixes

Basic monitoring means two things: **you hear about failures within a day**, and you have the numbers to see why.

BUILT INTO PROXMOX

- **Summary** of node and container: CPU, memory, network and disk graphs.
- **Node → Disks:** SMART health per drive. Shell: `smartctl -a /dev/nvme0`
- **Datacenter → Notifications:** add an email (SMTP), Gotify or webhook target, so a failed backup job reaches you.

INSIDE THE CONTAINER

```
ollama ps                      # loaded? CPU/GPU?
journalctl -e -u ollama        # Ollama logs
docker logs --tail 50 open-webui
free -h && df -h               # RAM and disk
```

WEEKLY 5-MINUTE CHECK

- ☐ Last night's backup job finished without errors
- ☐ Disk usage below about 80% on every storage
- ☐ Updates applied on host and container (p9)
- ☐ Last restore test less than a month ago (p10)

Troubleshooting

SYMPTOM	LIKELY CAUSE	FIX
Open WebUI lists no models	It cannot reach Ollama	Use `--network=host` with `OLLAMA_BASE_URL=http://127.0.0.1:11434` [17]
Answers suddenly much slower	Model spilled onto CPU, or into swap	Check `ollama ps` [9]; choose a smaller model or shorter context (p4)
Model fails to load	Not enough memory for model plus context	Smaller model, more container RAM, or less context [10]
Logged out after every update	No persistent secret key	Pass the key file from p8, step 2 [16]
`tailscale up` fails in the container	No `/dev/net/tun`	`pct set 101 --dev0 /dev/net/tun`, then restart the container [20]
Host updates fail with "401 Unauthorized"	Enterprise repo active without a subscription	Disable it and add no-subscription (p5, step 6)
Container won't start after an experiment	Broken config or package	`pct rollback 101 <snapshot>`, or restore a backup (p10)

Want uptime checks with phone alerts later? A self-hosted monitor such as Uptime Kuma is a popular next step. Keep it inside the tailnet, too.

11 Your 7-day starter plan

One focused session per day, roughly 1–2 hours (ROUGH ESTIMATE). Every day ends with a concrete **done when**. Missed a day? Shift the plan, don't skip day 6.

DAY 1 — **Hardware, firmware, Proxmox VE install** · p3, p5
Assemble, enable virtualization, write the USB stick, install with ext4 and a fixed IP.
DONE WHEN `https://<ip>:8006` shows the dashboard after a reboot.

DAY 2 — **Updates, two-factor login, first container** · p5, p6
Switch to the no-subscription repo, upgrade, enable 2FA, download the Debian 13 template, create CT 101.
DONE WHEN `pct enter 101` works and `apt update` runs clean inside.

DAY 3 — **Ollama and a first model** · p7
Install Ollama, run `llama3.2`, test the API, note your tokens/s with `--verbose`.
DONE WHEN the model answers and `ollama ps` shows where it runs.

DAY 4 — **Open WebUI** · p8
Docker from Debian, secret key file, start the container, create the admin account.
DONE WHEN you chat in the browser, and a second browser cannot sign up on its own.

DAY 5 — **Tailscale and the security checklist** · p9
TUN device, install, log in on all your devices; tick every box on the checklist.
DONE WHEN your phone on mobile data opens Open WebUI and your router has zero port forwards.

DAY 6 — **Backups and a real restore test** · p10
Backup storage, nightly job with retention, one manual backup, then restore it as CT 901.
DONE WHEN CT 901 booted from the backup and listed your models. Then destroy it.

DAY 7 — **Monitoring and right-sizing** · p4, p11
Notification target, SMART check, then pick your everyday model with the model ladder.
DONE WHEN a test notification arrived and you settled on a default model that fits with headroom.

AFTER WEEK ONE

Try one model size up · add a GPU (Tier 2) · point your own scripts at the local API [12] · move backups to a Proxmox Backup Server [1] · keep running the monthly restore test.

12 Short glossary

Proxmox VE – open-source hypervisor based on Debian; runs VMs and containers, with a web UI on port 8006.

VM – full virtual machine with its own kernel; needed for Windows and many GPU-passthrough setups.

Template – prebuilt OS image for new containers, managed with `pveam`.

Snapshot – point-in-time state you can roll back to; same disk, so not a backup.

3-2-1 – three copies, two types of media, one off-site.

Open WebUI – browser chat interface with user accounts, here on port 8080.

Quantization (4-bit, 8-bit) – storing weights with fewer bits to save memory, at some cost in quality.

Token – a chunk of text, often part of a word; speed is measured in tokens per second.

KV cache – memory holding the context while the model generates.

Memory bandwidth – how fast RAM or VRAM can be read; the main speed limit for text generation.

eGPU · OCuLink · USB4 – an external GPU and two common ways to connect one.

MagicDNS – Tailscale's automatic device names, e.g. `http://ai:8080`.

LXC container – lightweight Linux system that shares the host's kernel; low overhead.

Unprivileged – container whose root user maps to an ordinary user on the host; the safer default.

Nesting / keyctl – container features that let Docker and Tailscale run inside.

vzdump – Proxmox's backup tool for VMs and containers.

Ollama – runs local language models; CLI plus an API on port 11434.

Parameters (3B, 8B) – model size in billions of weights; bigger is usually smarter, larger and slower.

GGUF – file format for quantized models used by llama.cpp-based tools.

Context window – how many tokens the model considers at once; costs memory.

VRAM – memory on the graphics card; models that fit entirely run fastest.

MoE – mixture of experts; only part of the weights is used for each token.

Tailscale · tailnet – private network between your devices; a tailnet is your own such network.

Port forwarding – router rule exposing a device to the internet. This kit uses none.

FOLLOW THE BUILD

ZeroClaw documents a self-hosted AI stack in the open: local LLMs, a small Proxmox homelab, and what each change really costs. Same rules as this PDF: sources, no invented numbers.

All links · aipresentsig.github.io/zeroclaw
YouTube · @zeroclawbuild
Bluesky · @zeroclawbuild.bsky.social
Instagram · @zeroclaw.build
X · @zeroclaw_build

Spotted a mistake or an outdated command? Send a DM on any channel. This kit is versioned and gets fixed.

Provided as is, for learning. You are responsible for your own systems and data. Product names belong to their owners; no affiliation or sponsorship.

§ Sources

All pages accessed in October 2026. Commands follow these official docs; where we computed or estimated something, the text says so.

[1] Proxmox – Downloads (Proxmox VE 9.2-1 ISO, Proxmox Backup Server)
proxmox.com/downloads

[2] Proxmox VE Admin Guide – Installation: requirements, USB media, installer, web UI on :8006, ZFS memory
pve.proxmox.com/pve-docs/chapter-pve-installation.html

[3] Proxmox wiki – Upgrade from 8 to 9: PVE 9 repository files (deb822)
pve.proxmox.com/wiki/Upgrade_from_8_to_9

[4] Proxmox wiki – FAQ: Debian 13 base, CPU virtualization, application containers vs VMs
pve.proxmox.com/wiki/FAQ

[5] Proxmox VE – pct(1) manual: create, set, restore, snapshot, exec, destroy
pve.proxmox.com/pve-docs/pct.1.html

[6] Proxmox VE – pveam(1) manual: container templates
pve.proxmox.com/pve-docs/pveam.1.html

[7] Proxmox VE – vzdump(1) manual: modes, compression, retention
pve.proxmox.com/pve-docs-6/vzdump.1.html

[8] Ollama docs – Linux: install script, systemd service, logs, updates
docs.ollama.com/linux

[9] Ollama docs – FAQ: ollama ps, OLLAMA_HOST default, model location, keep-alive, local-only mode
docs.ollama.com/faq

[10] Ollama docs – Context length: VRAM-based defaults, memory note
docs.ollama.com/context-length

[11] Ollama docs – Hardware support: NVIDIA, AMD ROCm, Vulkan, Apple Metal
docs.ollama.com/gpu

[12] Ollama docs – API introduction: local base URL
docs.ollama.com/api/introduction

[13] Ollama README (v0.16.0): RAM guidance, model sizes for Llama 3.2 and Gemma 3
pkg.go.dev/github.com/ollama/ollama@v0.16.0

[14] Ollama library – qwen3 tags and download sizes
ollama.com/library/qwen3

[15] Google AI for Developers – Run Gemma with Ollama: gemma4 tags, image input
ai.google.dev/gemma/docs/integrations/ollama

[16] Open WebUI docs – Quick Start: Docker, secret key, admin account, updating, Python versions
docs.openwebui.com/getting-started/quick-start

[17] Open WebUI – TROUBLESHOOTING.md: --network=host and port 8080
github.com/open-webui/open-webui

[18] Debian – docker.io package (26.1.5 in Debian 13 "trixie")
packages.debian.org/src:docker.io

[19] Tailscale docs – Install Tailscale on Linux
tailscale.com/docs/install/linux

[20] Tailscale docs – Tailscale in LXC containers (TUN device, nesting, keyctl)
tailscale.com/kb/1130/lxc-unprivileged

[21] Tailscale docs – tailscale serve command; Serve vs Funnel
tailscale.com/docs/reference/tailscale-cli/serve

[22] Intel – Intel Processor N100 specifications (cores, memory channels, max memory)
intel.com · product SKU 231803

[23] NVIDIA Technical Blog – "Mastering LLM Techniques: Inference Optimization" (2023): memory-bound decoding, KV cache formula
developer.nvidia.com/blog/?p=73739

[24] U.S. Chamber of Commerce (CO—) – What Is the 3-2-1 Backup Rule (2025)
uschamber.com/co/run/technology/3-2-1-backup-rule

[25] Tailscale docs – Key expiry: 180-day default, disabling it for trusted servers
tailscale.com/docs/features/access-control/key-expiry

LABELS USED IN THIS KIT

RULE OF THUMB our approximation or arithmetic, derived from the cited sources · SOURCE taken directly from the cited page · **no benchmarks:** measure your own hardware with `ollama run --verbose`.