Hermes Agent

Hermes Agent installation, tutorials, configs.

- Hermes Agent on Debian 13 in a Proxmox VM

Hermes Agent on Debian 13 in a Proxmox VM

Hermes Agent on Debian 13 in a Proxmox VM

Verified scope

This guide installs **Hermes Agent by Nous Research**, not Hermes mail server, Hermes JS engine, or any other “Hermes” project. Hermes Agent is documented as an open-source autonomous agent that runs on Linux/macOS/WSL2/Termux and is installed with the upstream installer command from the official documentation (Hermes Agent Installation).

The target OS is **Debian 13 “trixie”** inside a Proxmox VM. Debian 13.0 was initially released on August 9, 2025, Debian 13.4 was current as of March 14, 2026, and amd64 is a supported architecture, which is the normal choice for a Proxmox VM on x86_64 hardware (Debian trixie release information).

The recommended installation mode here is:

- A dedicated non-root Linux user named `hermes`.
- Upstream Hermes installer, run as that non-root user.
- Optional Docker terminal backend for safer agent command execution.
- Optional Hermes gateway as a systemd user service or boot-time system service.
- Optional Caddy reverse proxy only for the API server, not for normal CLI/gateway bot usage.

Architecture

Hermes Agent stores its normal user configuration under `~/.hermes/config.yaml`, secrets under `~/.hermes/.env`, OAuth credentials under `~/.hermes/auth.json`, logs under `~/.hermes/logs/`, sessions under `~/.hermes/sessions/`, memories under `~/.hermes/memories/`, skills under `~/.hermes/skills/`, and cron jobs under `~/.hermes/cron/` (Hermes Agent Configuration).

The upstream installer’s documented prerequisite is Git, and the installer handles uv, Python 3.11, Node.js v22, ripgrep, ffmpeg, the repository clone, virtual environment, global `hermes` command

setup, and LLM provider configuration (Hermes Agent Installation).

On Debian 13, the distro `python3` package tracks Python 3.13, while Hermes' installer manages Python 3.11 through uv, so do not force Hermes to use Debian's default Python unless you have a development reason to do so (Debian python3 package, Hermes Agent Installation).

On Debian 13, the distro `nodejs` package is Node.js 20.19.2, while Hermes' installer downloads Node.js v22 into the Hermes user environment for browser automation and WhatsApp bridge needs, so the guide avoids installing Debian's `nodejs` as the primary Hermes runtime (Debian nodejs package, Hermes Agent Installation).

Hermes requires at least one inference provider, and the official provider setup path is `hermes model` from the terminal, not `/model` inside an existing chat session (Hermes Agent AI Providers).

Hermes Agent requires a model with at least **64,000 tokens** of context according to the official quickstart, and local/self-hosted model users must configure the model server with enough context rather than only setting a client-side value (Hermes Agent Quickstart).

Proxmox VM sizing

Use this as a practical baseline:

Purpose	vCPU	RAM	Disk	Notes
CLI only, hosted LLM provider	2	4 GB	32 GB	Good for OpenRouter, Nous Portal, Anthropic, Gemini, Copilot, etc.
Gateway service plus Docker backend	4	8 GB	64 GB	Better for always-on bot usage and isolated terminal backend.
Local model in same VM	8+	32 GB+	150 GB+	Only if you attach GPU/large RAM; otherwise run Ollama/vLLM elsewhere.

These are operational recommendations, not upstream minimums. Hermes' official docs describe the installer and runtime providers, while actual VM sizing depends on whether the LLM runs remotely or inside the VM (Hermes Agent Installation, Hermes Agent AI Providers).

Recommended Proxmox VM hardware:

- BIOS: OVMF/UEFI or SeaBIOS; either is fine for Debian 13.

- Machine: q35.
- CPU type: `host` if migration compatibility is not required; otherwise use your cluster baseline.
- Disk bus: SCSI with VirtIO SCSI.
- Network: VirtIO NIC on the correct bridge, usually `vmbr0`.
- Guest agent: enable in Proxmox and install `qemu-guest-agent` inside Debian.
- Backups: enable Proxmox backups after the guest agent works.
- Snapshot: take a clean snapshot before running the Hermes installer.

The QEMU guest agent is used by Proxmox for better guest shutdown and filesystem freeze/thaw during backup/snapshot operations, so it is worth installing before treating the VM as production (Proxmox QEMU Guest Agent).

1. Create the Debian 13 VM

1. Download the current Debian 13 amd64 netinst ISO from Debian's official download path.
2. In Proxmox, create a VM with the sizing above.
3. Select Debian/Linux as the guest OS type.
4. Use VirtIO network and SCSI/VirtIO disk.
5. Install only the SSH server and standard system utilities unless you need a desktop.
6. Set timezone to `Europe/Athens` if this VM is for your local environment.
7. Reboot into the installed system.

After first boot, SSH into the VM as your administrative user.

```
ssh your_admin_user@VM_IP
```

Verify Debian release:

```
cat /etc/os-release
uname -a
```

Expected result:

- `PRETTY_NAME` should show Debian GNU/Linux 13 or trixie.
- Kernel and architecture should be normal for your Proxmox environment.

2. Update Debian and install base packages

Run:

```
sudo apt update
sudo apt full-upgrade -y
sudo apt install -y \
  ca-certificates \
```

```
  curl \
  git \
  gnupg \
  lsb-release \
  qemu-guest-agent \
  sudo \
  tmux \
  unzip \
  vim \
  wget
sudo systemctl enable --now qemu-guest-agent
sudo reboot
```

The upstream Hermes documentation says Git is the only prerequisite, but this guide installs common server tools and `qemu-guest-agent` so the VM behaves correctly under Proxmox (Hermes Agent Installation, Proxmox QEMU Guest Agent).

After the reboot, verify the guest agent from the Proxmox host:

```
qm agent VMID ping
```

Proxmox documents `qm agent <vmid> ping` as the test command for QEMU guest-agent communication (Proxmox QEMU Guest Agent).

3. Enable guest agent in Proxmox

If you did not enable it in the GUI, run this on the Proxmox host:

```
qm set VMID --agent 1
```

Replace `VMID` with the VM ID. Proxmox documents `qm set VMID --agent 1` as the host-side enable command for QEMU guest-agent communication (Proxmox QEMU Guest Agent).

4. Create a dedicated Hermes user

Create a non-root user:

```
sudo adduser hermes
sudo usermod -aG sudo hermes
```

For a hardened always-on gateway, remove passwordless sudo if your template added it and use sudo only when you intentionally administer the system. Hermes' own security docs explicitly

recommend running the gateway as non-root (Hermes Agent Security).

Switch to the Hermes user:

```
sudo -iu hermes
```

Verify:

```
whoami
pwd
```

Expected:

```
hermes
/home/hermes
```

5. Install Hermes Agent

Run the official upstream installer as the `hermes` user:

```
curl -fsSL https://raw.githubusercontent.com/NousResearch/hermes-agent/main/scripts/install.sh
| bash
```

This is the official Linux/macOS/WSL2/Termux install command in the Hermes Agent installation docs (Hermes Agent Installation).

Reload the shell:

```
source ~/.bashrc
```

Verify the command exists:

```
command -v hermes
hermes --help
```

The official post-install instruction is to reload the shell and start Hermes with `hermes` (Hermes Agent Installation).

6. Verify installer layout

Run:

```
ls -la ~/.hermes
ls -la ~/.hermes/hermes-agent
ls -la ~/.local/bin/hermes
```

For non-root installs, the upstream installer defaults to code under `~/.hermes/hermes-agent` and links the `hermes` command under `~/.local/bin`; this behavior is visible in the official installer script (Hermes install.sh on GitHub).

Expected important files:

```
~/.hermes/config.yaml
~/.hermes/.env
~/.hermes/logs/
~/.hermes/sessions/
~/.hermes/memories/
~/.hermes/skills/
```

These paths match the official Hermes configuration directory structure (Hermes Agent Configuration).

Secure the secrets file:

```
chmod 600 ~/.hermes/.env
```

Hermes' security documentation explicitly recommends `chmod 600 ~/.hermes/.env` for API keys and secrets (Hermes Agent Security).

7. Run diagnostics

Run:

```
hermes doctor
hermes config check
```

`hermes doctor` is the official diagnostic command, and `hermes config check` checks for missing options after installs or updates (Hermes Agent Installation, Hermes Agent Configuration).

If config migration is requested:

```
hermes config migrate
```

`hermes config migrate` is the official command to add missing configuration options after updates (Hermes Agent Configuration).

8. Configure the model provider

Run the provider setup wizard from the terminal:

```
hermes model
```

The official provider docs say `hermes model` is the full setup wizard for OAuth, API keys, provider selection, and custom endpoints, while `/model` inside a chat only switches among already configured providers (Hermes Agent AI Providers).

Recommended choices:

Scenario	Provider path
Easiest hosted setup	Nous Portal or OpenRouter
You already use Claude	Anthropic / Claude Code auth
You already use GitHub Copilot	GitHub Copilot
You want local/private inference	Custom endpoint to Ollama, vLLM, SGLang, llama.cpp, or LiteLLM
You want provider routing	OpenRouter

Hermes supports providers including Nous Portal, OpenAI Codex, GitHub Copilot, Anthropic, OpenRouter, Google/Gemini, DeepSeek, Hugging Face, Ollama Cloud, AWS Bedrock, and custom OpenAI-compatible endpoints (Hermes Agent AI Providers).

OpenRouter example

Use the wizard:

```
hermes model
```

Or set the key directly:

```
hermes config set OPENROUTER_API_KEY sk-or-v1-REPLACE_ME
hermes model
```

The official docs show API provider keys stored in `~/.hermes/.env`, and `hermes config set OPENROUTER_API_KEY ...` saves the secret in the right place (Hermes Agent Configuration).

Local Ollama example

If Ollama runs on the same VM:

```
ollama pull qwen2.5-coder:32b
OLLAMA_CONTEXT_LENGTH=65536 ollama serve
```

Then configure Hermes:

```
hermes model
# Select: Custom endpoint (self-hosted / VLLM / etc.)
# URL: http://localhost:11434/v1
# API key: leave empty or use ollama
# Model: qwen2.5-coder:32b
# Context length: 65536
```

The official provider docs describe local Ollama as an OpenAI-compatible custom endpoint at `http://localhost:11434/v1` and warn that context must be configured server-side for Ollama (Hermes Agent AI Providers).

Verify Ollama context:

```
ollama ps
```

The official provider docs say to check the `CONTEXT` column in `ollama ps` to verify the effective context length (Hermes Agent AI Providers).

Important: Hermes' quickstart says models below 64,000 tokens of context are rejected at startup, while the providers page also explains practical local-agent context issues around smaller context windows; for Hermes Agent on a server, target 64K context unless you have verified a newer upstream behavior in your installed version (Hermes Agent Quickstart, Hermes Agent AI Providers).

9. First chat test

Run:

```
hermes
```

Send:

```
Reply with exactly: HERMES_OK
```

Success criteria:

- Hermes starts without a Python, Node, or provider error.

- The banner shows the selected provider/model.
- The response contains `HERMES_OK`.

The official quickstart says success looks like a banner showing the chosen model/provider, a normal reply, and successful multi-turn operation (Hermes Agent Quickstart).

Exit, then test session persistence:

```
hermes --continue
```

`hermes --continue` and `hermes -c` are the official commands to resume the most recent session (Hermes Agent Quickstart).

10. Configure safer terminal execution

For a Proxmox server, use Docker backend unless you explicitly want Hermes to run commands directly on the VM. Hermes' security docs state that local backend has no isolation, while Docker, Singularity, Modal, and Daytona provide isolation boundaries for terminal execution (Hermes Agent Security).

Install Docker from Debian:

```
exit
sudo apt update
sudo apt install -y docker.io
sudo systemctl enable --now docker
sudo usermod -aG docker hermes
sudo -iu hermes
newgrp docker
docker run --rm hello-world
```

Debian documents `docker.io` as Debian's Docker package, and Debian 13 provides `docker.io` for trixie (Debian Docker wiki, Debian docker.io package).

Set Hermes terminal backend:

```
hermes config set terminal.backend docker
```

The Hermes configuration docs show `hermes config set terminal.backend docker` as the command to switch the terminal backend to Docker (Hermes Agent Configuration).

Review Docker resource defaults:

```
hermes config edit
```

Recommended Docker block:

```
terminal:
  backend: docker
  docker_image: "nikolaik/python-nodejs:python3.11-nodejs20"
  docker_forward_env: []
  container_cpu: 2
  container_memory: 5120
  container_disk: 51200
  container_persistent: true
```

Hermes' security docs recommend Docker, Modal, or Daytona for production gateway deployments and show `docker_forward_env: []` to keep secrets out of the container unless explicitly allowed (Hermes Agent Security).

Run a tool-capable chat test:

```
hermes
```

Ask:

```
Use the terminal to run: pwd && whoami && python --version
```

Expected:

- The command runs in the configured terminal backend.
- If Docker backend is active, the execution environment is containerized.
- Dangerous commands still require attention depending on backend and approval settings.

11. Configure command approval behavior

Open config:

```
hermes config edit
```

Recommended:

```
approvals:
  mode: manual
  timeout: 60
```

`approvals.mode` supports `manual`, `smart`, and `off`, and the default manual mode prompts for dangerous commands while timeout denial fails closed (Hermes Agent Security).

Do not use this on a server unless you fully understand the risk:

```
approvals:
  mode: off
```

Hermes documents `approvals.mode: off`, `hermes --yolo`, `/yolo`, and `HERMES_YOLO_MODE=1` as ways to bypass dangerous command approvals, and warns that disabling approval checks is equivalent to running with `--yolo` (Hermes Agent Security).

12. Optional gateway setup

The gateway is the always-on background process that connects Hermes to messaging platforms, handles sessions, runs cron jobs, and delivers voice messages (Hermes Agent Messaging Gateway).

Configure platforms:

```
hermes gateway setup
```

Start in foreground first:

```
hermes gateway
```

The official gateway docs list `hermes gateway setup` for interactive platform configuration and `hermes gateway` for foreground execution (Hermes Agent Messaging Gateway).

In another SSH session:

```
sudo -iu hermes
hermes gateway status
tail -f ~/.hermes/logs/gateway.log
```

The official docs list `hermes gateway status` and gateway log paths for checking the service (Hermes Agent Messaging Gateway).

13. Gateway allowlists

Edit secrets:

```
nano ~/.hermes/.env
```

Set only the platforms you use:

```
# Telegram example
TELEGRAM_BOT_TOKEN=REPLACE_ME
TELEGRAM_ALLOWED_USERS=123456789

# Discord example
DISCORD_TOKEN=REPLACE_ME
DISCORD_ALLOWED_USERS=111222333444555666

# Global allowlist fallback
GATEWAY_ALLOWED_USERS=123456789
```

Hermes' gateway docs recommend restricting access to specific users and show platform-specific allowlist variables such as `TELEGRAM_ALLOWED_USERS`, `DISCORD_ALLOWED_USERS`, and `GATEWAY_ALLOWED_USERS` (Hermes Agent Messaging Gateway).

Do not use this in production:

```
GATEWAY_ALLOW_ALL_USERS=true
```

Hermes explicitly marks `GATEWAY_ALLOW_ALL_USERS=true` as not recommended for bots with terminal access, and the security docs state the default behavior denies users who are neither allowlisted nor paired (Hermes Agent Messaging Gateway, Hermes Agent Security).

Secure `.env` again:

```
chmod 600 ~/.hermes/.env
```

14. Install gateway as a service

Option A: systemd user service

Run as `hermes`:

```
hermes gateway install
hermes gateway start
hermes gateway status
journalctl --user -u hermes-gateway -f
```

The official gateway docs list `hermes gateway install`, `start`, `status`, and `journalctl --user -u hermes-gateway -f` for a Linux user service (Hermes Agent Messaging Gateway).

Keep the user service running after logout:

```
exit
sudo loginctl enable-linger hermes
```

The official gateway docs recommend `sudo loginctl enable-linger $USER` to keep a user service running after logout (Hermes Agent Messaging Gateway).

Option B: boot-time system service

Run from the `hermes` account if it has sudo:

```
sudo hermes gateway install --system
sudo hermes gateway start --system
sudo hermes gateway status --system
journalctl -u hermes-gateway -f
```

The official gateway docs list `sudo hermes gateway install --system`, `start --system`, `status --system`, and `journalctl -u hermes-gateway -f` for a Linux boot-time system service (Hermes Agent Messaging Gateway).

15. Optional API server

Only enable the API server if you need Hermes exposed as an OpenAI-compatible HTTP backend for another frontend. The Hermes API server exposes the agent as an OpenAI-compatible endpoint and gives access to the agent's full toolset, including terminal commands, so treat it as sensitive (Hermes Agent API Server).

Edit:

```
nano ~/.hermes/.env
```

Add:

```
API_SERVER_ENABLED=true
API_SERVER_HOST=127.0.0.1
API_SERVER_PORT=8642
API_SERVER_KEY=replace-with-a-long-random-token
```

The API server docs show `API_SERVER_ENABLED`, `API_SERVER_HOST`, `API_SERVER_PORT`, and `API_SERVER_KEY`, with default host `127.0.0.1` and default port `8642` (Hermes Agent API Server).

Restart gateway:

```
hermes gateway restart || {
  hermes gateway stop
  hermes gateway start
}
```

Test locally:

```
curl http://localhost:8642/v1/chat/completions \
  -H "Authorization: Bearer replace-with-a-long-random-token" \
  -H "Content-Type: application/json" \
  -d '{"model":"hermes-agent","messages":[{"role":"user","content":"Reply with API_OK"}]}'
```

The official API server docs show the same `/v1/chat/completions` test pattern with Bearer-token authorization (Hermes Agent API Server).

16. Optional Caddy reverse proxy

Keep the Hermes API bound to `127.0.0.1` and let Caddy handle TLS. This avoids binding Hermes directly to the network while still allowing HTTPS access through Caddy.

Example Caddyfile:

```
hermes.example.com {
  reverse_proxy 127.0.0.1:8642
}
```

Do **not** expose the API server without a strong `API_SERVER_KEY`, because Hermes' API server docs state that Bearer-token auth is required when binding to non-loopback addresses and that the endpoint exposes the full toolset including terminal commands (Hermes Agent API Server).

If a browser app calls the API directly, set CORS explicitly:

```
API_SERVER_CORS_ORIGINS=https://your-frontend.example.com
```

The API server docs describe `API_SERVER_CORS_ORIGINS` as the comma-separated allowlist for browser origins (Hermes Agent API Server).

17. Firewall rules

For normal Telegram/Discord/Slack/etc. gateway use, the VM usually needs outbound Internet access only, because bot adapters typically connect outward to platform APIs.

Minimum inbound firewall:

```
sudo apt install -y nftables
sudo systemctl enable --now nftables
```

Example `/etc/nftables.conf`:

```
#!/usr/sbin/nft -f

flush ruleset

table inet filter {
  chain input {
    type filter hook input priority 0;
    policy drop;

    iif "lo" accept
    ct state established,related accept

    tcp dport 22 accept

    # Optional: only if Caddy is installed on this VM
    tcp dport {80, 443} accept

    icmp type echo-request accept
    ip6 nexthdr ipv6-icmp accept
  }

  chain forward {
    type filter hook forward priority 0;
    policy drop;
  }

  chain output {
    type filter hook output priority 0;
    policy accept;
  }
}
```

Apply:

```
sudo nft -f /etc/nftables.conf
sudo nft list ruleset
```

This firewall section is an operational baseline for the VM; Hermes-specific exposure decisions are governed by the gateway and API server configuration described in the official docs (Hermes Agent Messaging Gateway, Hermes Agent API Server).

18. Backups

Back up at least:

```
/home/hermes/.hermes/
```

That directory contains config, secrets, OAuth credentials, sessions, memories, skills, cron jobs, and logs according to the official directory layout (Hermes Agent Configuration).

Suggested local backup script:

```
sudo install -d -m 700 -o hermes -g hermes /home/hermes/backups
sudo -iu hermes bash -lc '
set -euo pipefail
ts="$(date -u +%Y%m%d-%H%M%S)"
tar --exclude="$HOME/.hermes/hermes-agent/.git" \
    -czf "$HOME/backups/hermes-home-$ts.tar.gz" \
    -C "$HOME" .hermes
ls -lh "$HOME/backups/hermes-home-$ts.tar.gz"
'
```

For Proxmox backups, make sure QEMU guest agent works before relying on snapshot consistency, because Proxmox uses the guest agent to freeze and thaw guest filesystems during backup/snapshot operations (Proxmox QEMU Guest Agent).

19. Update procedure

Run as `hermes`:

```
hermes update
hermes config check
hermes config migrate
hermes doctor
```

`hermes update` is listed in the official quick reference, and `hermes config check` plus `hermes config migrate` are the documented post-update configuration checks (Hermes Agent Quickstart, Hermes Agent Configuration).

If the gateway runs as a service:

```
hermes gateway restart
hermes gateway status
```

The gateway docs include start/stop/status service management commands, and restart is listed in Hermes gateway command references in the docs (Hermes Agent Messaging Gateway).

20. Rollback procedure

Before major changes:

1. Stop the gateway.
2. Take a Proxmox snapshot.
3. Back up `/home/hermes/.hermes/`.
4. Run updates.
5. Verify with `hermes doctor`, a CLI chat, and gateway status.

Stop service:

```
sudo -iu hermes hermes gateway stop || true
```

Restore local Hermes state:

```
sudo -iu hermes
mv ~/.hermes ~/.hermes.broken.$(date -u +%Y%m%d-%H%M%S)
tar -xzf ~/backups/hermes-home-YYYYMMDD-HHMMSS.tar.gz -C ~
chmod 600 ~/.hermes/.env
hermes doctor
```

If the VM itself is broken, restore the Proxmox snapshot or backup.

21. Verification checklist

Run these in order:

```
cat /etc/os-release
git --version
command -v hermes
```

```
hermes --help
hermes doctor
hermes config check
hermes config show
```

Expected:

- Debian reports version 13/trixie.
- Git exists.
- `hermes` resolves from `/home/hermes/.local/bin/hermes`.
- `hermes doctor` does not report blocking errors.
- `hermes config check` does not require unresolved migrations.

Provider test:

```
hermes chat -q "Reply with exactly PROVIDER_OK"
```

Gateway test:

```
hermes gateway status
journalctl --user -u hermes-gateway -n 100 --no-pager
```

API server test if enabled:

```
curl -fsS http://localhost:8642/v1/models \
  -H "Authorization: Bearer replace-with-a-long-random-token"
```

Docker backend test if enabled:

```
docker info >/dev/null
hermes config show | grep -i 'backend'
```

22. Troubleshooting

`hermes: command not found`

Run:

```
source ~/.bashrc
echo "$PATH"
ls -la ~/.local/bin/hermes
```

The official installation troubleshooting says to reload the shell or check PATH when `hermes: command not found` appears (Hermes Agent Installation).

API key not set

Run:

```
hermes model
```

Or set the key:

```
hermes config set OPENROUTER_API_KEY sk-or-v1-REPLACE_ME
```

The official installation troubleshooting says `API key not set` is fixed by running `hermes model` or setting the provider key through config (Hermes Agent Installation).

Missing config after update

Run:

```
hermes config check
hermes config migrate
```

The official installation troubleshooting lists this exact fix for missing config after update (Hermes Agent Installation).

Gateway starts but nobody can message it

Run:

```
hermes gateway setup
hermes gateway status
grep -E 'ALLOWED_USERS|ALLOW_ALL|TOKEN' ~/.hermes/.env
```

The official quickstart says gateway message failures are usually caused by bot token, allowlist, or incomplete platform setup, and recommends re-running `hermes gateway setup` plus checking `hermes gateway status` (Hermes Agent Quickstart).

Local model “works” but Hermes fails or gives bad tool output

Verify:

```
curl http://localhost:11434/v1/models
ollama ps
hermes model
```

The provider docs warn that wrong base URL, model name, endpoint compatibility, or context configuration can break custom endpoints, and they give `curl http://localhost:11434/v1/models` and `ollama ps` as verification steps (Hermes Agent AI Providers).

Context length errors

Check the startup line or run `/usage` inside a session, then set an explicit context length:

```
model:
  context_length: 65536
```

The FAQ says the CLI shows detected context length, `/usage` shows usage during a session, and `model.context_length` can override the detected model context (Hermes Agent FAQ).

Docker permission denied

Run:

```
sudo usermod -aG docker hermes
sudo -iu hermes
newgrp docker
docker run --rm hello-world
```

The Hermes FAQ lists `docker info`, adding the user to the `docker` group, `newgrp docker`, and `docker run hello-world` as Docker troubleshooting commands (Hermes Agent FAQ).

Gateway logs

User service:

```
journalctl --user -u hermes-gateway -f
```

System service:

```
journalctl -u hermes-gateway -f
```

Hermes' gateway docs list these exact journalctl commands for user and system services (Hermes Agent Messaging Gateway).

Final hardening checklist

- The VM has Proxmox QEMU guest agent installed, enabled, and pingable.
- The `hermes` Linux user owns `/home/hermes/.hermes`.
- `~/.hermes/.env` is mode `600`.
- `GATEWAY_ALLOW_ALL_USERS=true` is not set.
- Platform-specific or global allowed-user lists are set for every messaging platform.
- `terminal.backend` is `docker` for always-on gateway use.
- Docker resource limits are configured.
- `approvals.mode` is `manual` unless running inside a deliberately disposable sandbox.
- API server is disabled unless needed.
- API server stays bound to `127.0.0.1` unless there is a strong Bearer token and firewall/reverse-proxy policy.
- Proxmox backups include the VM and Hermes state.
- A manual backup of `/home/hermes/.hermes/` exists before upgrades.

This hardening checklist follows Hermes' own production gateway recommendations: explicit allowlists, container backend, resource limits, secure secrets, DM pairing when useful, command allowlist review, safe working directory, non-root gateway, log monitoring, and regular updates (Hermes Agent Security).